National Information Exchange Model Model Package Description Specification

Version 3.0alpha1

August 30, 2013

NIEM Technical Architecture Committee (NTAC)

Contents

Authors

Mark Kindl, Georgia Tech Research Institute

Abstract

This document specifies normative rules and non-normative guidance for building Model Package Descriptions (MPDs) that conform to the National Information Exchange Model (NIEM) version 3.0.

Status

This document is a draft of the specification for NIEM Model Package Descriptions (MPDs). It represents the design that has evolved from the collaborative work of the NIEM Business Architecture Committee (NBAC) and the NIEM Technical Architecture Committee (NTAC) and their predecessors.

This specification is a product of the NIEM Program Management Office (PMO).

Send comments on this specification via email to `niem-comments@lists.gatech.edu`.

1. Introduction

This specification assumes familiarity with the National Information Exchange Model (NIEM), its basic concepts, architecture, processes, design rules, and general conformance rules. For novices, the recommended reading list includes:

- Introduction to the National Information Exchange Model **[NIEM Introduction]**
- NIEM Concept of Operations **[NIEM Concept of Operations]**
- NIEM Naming and Design Rules **[NIEM NDR]**
- NIEM High-Level Version Architecture **[NIEM High-Level Version Architecture]**
- NIEM High-Level Tool Architecture **[NIEM High-Level Tool Architecture]**
- NIEM Conformance **[NIEM Conformance]**
- NIEM User Guide **[NIEM User Guide]**
- NIEM Business Information Exchange Components (BIEC) **[NIEM BIEC]**
- NIEM Implementation Guidelines **[NIEM Implementation Guidance]**

The foregoing NIEM documents are available at `http://reference.niem.gov/niem/`. See **[NIEM Implementation Guidance]** for NIEM Implementation Guidelines.

Those knowledgeable of NIEM should be familiar with the **[NIEM NDR]**, **[NIEM High-Level Version Architecture]**, **[NIEM Conformance]**, and **[NIEM BIEC]**.

This specification uses and is a peer to the NIEM Naming and Design Rules (NDR) **[NIEM NDR]** and supersedes IEPD guidance previously published in Requirements for a NIEM IEPD **[NIEM IEPD Requirements]** and the NIEM User Guide **[NIEM User Guide]**. The NIEM User Guide remains a good source for understanding the process of building Information Exchange Package Documentation (IEPD).

1.1. Background

Many fundamental concepts, processes, and products in the NIEM generally involve aggregating electronic files into logical sets that serve a specific purpose. Examples of such sets include, but in the future may not necessarily be limited to, a NIEM release, core update (CU), domain update (DU), Information Exchange Package Documentation (IEPD), and Enterprise Information Exchange Model (EIEM). Each of these examples is a NIEM Model Package Description (MPD).

[Definition: Model Package Description (MPD)]

A set of related W3C XML Schema documents and other supporting files organized as one of the five classes of NIEM schema sets:

- Release (major, minor, or micro).
- Domain update (DU) to a release.
- Core update (CU) to a release.
- Information Exchange Package Documentation (IEPD).
- Enterprise Information Exchange Model (EIEM).

An MPD is self-documenting and provides sufficient normative and non-normative information to allow technical personnel to understand how to use or implement it. An MPD is packaged as a ZIP **[PKZIP]** file.

A key NIEM concept used throughout this specification is *data component*.

[Definition: data component]

An XML Schema type or attribute group definition; or an XML Schema element or attribute declaration.

A MPD is a normative specification for XML data components in the format of World Wide Web Consortium (W3C) XML Schema definition language **[W3 XML Schema Datatypes]**, **[W3 XML Schema Structures]**. MPD schema documents either (1) define the semantics and structure for NIEM reusable data components, or (2) define implementable NIEM exchange document instances in W3C Extensible Markup Language (XML) **[W3-XML]**.

A NIEM MPD is complete when it has been properly packaged with the schemas, documentation, and supplemental files needed to understand how to use and implement it. MPD content design, development, and assembly may be difficult and time-consuming, especially if done manually. Software tools have been shown to significantly reduce the complexity of designing, constructing, changing, and managing MPDs. In order to reduce ambiguity and to facilitate interoperable and effective tool support, this baseline specification imposes some degree of consistency on the terminology, syntax, semantics, and composition of MPDs.

1.2. Purpose

This document is a normative specification for the various kinds of NIEM MPDs. The rules and guidance herein are designed to encourage and facilitate NIEM use and tools by balancing consistency, simplicity, and flexibility. Consistency and simplicity make MPDs easy to design correctly, build rapidly, and find easily (for reuse or adaptation). Consistency also facilitates tool support. Flexibility enables more latitude to design and tailor MPDs for complex data exchange requirements. As such, this document does not necessarily prescribe mandates or rules for all possible situations or organizational needs. If an organization determines it should impose additional constraints or requirements on its IEPDs beyond those specified in this document (for example, mandating a normative set of business requirements or a domain model within IEPD documentation), then it is free to do so, as long as no conflicts exist with this MPD Specification or the **[NIEM NDR]**.

This document defines terminology; identifies required and optional (but common) artifacts; defines metadata; specifies normative rules, schemes, and syntax; provides non-normative guidance; and as needed, refers to other related NIEM specifications for more detail.

1.3. Scope

This specification applies to information exchange definitions and release products that employ the data component definitions and declarations in NIEM Core and Domains. In particular, this version of this document applies to the following MPDs:

- NIEM releases (including major, minor, and micro releases).
- NIEM domain updates (DU) **[NIEM Domain Update Specification]**. (Note these are NOT the same as the NIEM domain schemas that are part of numbered releases).
- Core updates (CU) to a release.
- Information Exchange Package Documentation (IEPD) that define NIEM data exchanges.
- Enterprise Information Exchange Model (EIEM) from which one or more NIEM IEPDs can be built or based.

In the future, as required, other types of MPDs may be added to this list.

At any point in time, an incomplete MPD will be in some state of development. This specification is applicable to such developing products in that it establishes standards on the final, published, production-quality state. In turn, tool vendors can craft, adapt, and/or integrate software tools that will assist in the development of MPDs from raw parts to finished product.

NIEM is a data layer for an information architecture. Files in an MPD generally define XML Schema types and declare XML elements and attributes to use in payloads for information exchanges. While an MPD may also contain files from layers beyond the data layer, this specification is not intended to define details of other architectural layers. Such files are generally present only to provide additional context, understanding, or assistance for implementing the exchange of payloads.

Authoritative sources are not required to revise MPDs that exist before this specification becomes effective. However, they are always encouraged to consider revising MPDs to meet this specification, especially when making other significant changes.

1.4. Audience

The following groups should review and adhere to this specification:

- The NIEM release manager who is responsible to integrate and publish NIEM releases and core updates.
- NIEM domain stewards and technical representatives who develop and publish domain updates.

- NIEM IEPD developers and implementers.
- NIEM tool developers and vendors.
- Organizations that intend to develop an EIEM.
- Individuals or groups responsible to review and approve MPDs.

2. Concepts and Terminology

The presentation of concepts and terms in this section is sequenced for understanding. Each subsection builds upon previous ones. This section concludes with an explanation of each of the five MPD classes and a summary of their similarities and differences.

2.1. Key Words for Requirement Levels

Within normative content rules and definitions, the key words MUST, MUST NOT, SHALL, SHALL NOT, SHOULD, SHOULD NOT, MAY, RECOMMENDED, REQUIRED, and OPTIONAL in this document are to be interpreted as described in **[RFC2119 Key Words]**.

2.2. Character Case Sensitivity

This specification imposes many constraints on the syntax for identifiers, names, labels, strings, etc. In all cases, unless otherwise explicitly noted, syntax is case sensitive. In particular, XML files in appendices that define particular artifacts, transformations, and examples are case sensitive.

Also, note that as a general principle, lower case characters are used whenever such will not conflict with the **[NIEM NDR]**.

2.3. Artifacts

MPDs are generally composed of files and file sets grouped for a particular purpose. Each file AND each logical set of such files is called an artifact. In other words, we refer to a set of files (with a defined purpose) as an artifact, and we refer to each file within that set as an artifact.

[Definition: artifact]

> A single file with a defined purpose or a set of files logically grouped for a defined purpose. An MPD is a collection of artifacts, the purpose for which is to define and document the intended use of the MPD.

While the kernel of an MPD is its XML schema document (XSD) artifacts, there are also other kinds of MPD artifacts. These may include HTML (or XML converted to HTML for display), text, or graphic files used for human-readable documentation. An MPD may also have artifacts intended to help assist in or accelerate the use and implementation of the MPD. For example, these may be XML, UML, or binary files that are inputs to or outputs from software tools used to build, generate, or edit the MPD or its schema artifacts. Appendix F: MPD Artifacts contains a listing of mandatory and common optional artifacts for the five types of MPDs. The various types of artifacts are described in more detail in subsequent sections.

2.4. Schema-Namespace Correspondence in NIEM

To simplify automatic schema processing and reduce the potential for confusion and error, **[NIEM NDR]** Principle 8 states that each NIEM-conformant namespace SHOULD be defined by exactly one reference schema document. To support this principle, **[NIEM NDR]** Rule 6-5 disallows the use of `xs:include`, and **[NIEM NDR]** Rule 6-35 mandates the use of the `xs:schema targetNamespace` attribute in NIEM-conformant schema documents.

The foregoing NDR rules and principle imply that each NIEM namespace is a single NIEM-conformant schema document, and each NIEM-conformant schema document declares a target namespace. NIEM does not permit schema documents without target namespaces, unless they are from sources outside of NIEM.

2.5. Harmonization

Harmonization is a process that NIEM governance committees and domain stewards iteratively apply to NIEM content (specifically, its semantics, structure, and relationships) during the preparation of a NIEM major or minor release. The result is change and evolution of the model with the intent of removing semantic duplication and overlap while improving representational quality and usability.

[Definition: harmonization]

> Given a data model, harmonization is the process of reviewing its existing data definitions and declarations; reviewing how it structures and represents data; integrating new data components; and refactoring data components as necessary to remove (or reduce to the maximum extent) semantic duplication and/or semantic overlap among all data structures and definitions resulting in representational quality improvements.

2.6. Validation

This specification often refers to the process of validation, that is, validation of XML schemas and XML instances. Generally, this should occur periodically during and after design time to ensure the conformance and quality of an information exchange definition. However, local architecture or policy may also dictate the need to validate more often, and in some cases may require runtime validation.

XML schema document sets that define a NIEM information exchange must be authoritative. Application developers may use other schemas (e.g., constraint schema documents) for various purposes, but for the purposes of determining NIEM conformance, the authoritative schemas are relevant. This does not mean that XML validation must be performed on all XML instances as they are served or consumed; only that the XML instances validate if and when XML validation is performed. Therefore, even when validation is not performed, instance XML documents must be valid against the XML schema document sets that specify them.

2.7. Reference Schema Document

A NIEM reference schema document is a schema document that is intended to be the authoritative definition of business semantics for components within its target namespace. Reference schema documents include the NIEM Core schema documents, NIEM domain schema documents, and NIEM domain update schema documents. The normative definition for a reference schema document and applicable conformance rules are found in the **[NIEM NDR]**. The definition is repeated here:

[Definition: reference schema document]

An XML Schema document that meets all of the following criteria:

- It is a conformant schema document.
- It is explicitly designated as a reference schema document. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema document.
- It provides the broadest, most fundamental definitions of components in its namespace.
- It provides the authoritative definition of business semantics for components in its namespace.
- It is intended to serve as the basis for components in IEPD and EIEM schema documents, including subset, constraint, extension, and exchange schema documents.

See also **reference schema document set**.

[Definition: reference schema document set]

A set of related reference schema documents, such as a NIEM release. See also **reference schema document**.

The **[NIEM NDR]** conformance rules for reference schema documentss are generally stricter than those for other classes of NIEM-conformant schema documents. For example, they are required to employ an `xs:annotation` with `xs:documentation`, and `xs:appinfo` elements that encapsulate semantic information for each XML element and attribute declaration, and type definition.

Reference schemas are very uniform in their structure. As they are the primary definitions for data components, they do not need to restrict other data definitions, and they are not allowed to use XML Schema's complex type restriction mechanisms.

2.8. Coherence of Schema Sets

A NIEM release is always a coherent set of reference schema documents in which multiple versions of semantically identical types or properties do not exist; and all types and properties are uniquely defined and declared. Each numbered release has been harmonized, tested, and carefully reviewed by NIEM governance committees in order to eliminate semantic duplication. The **[NIEM High-Level Version Architecture]** defines a coherent schema document set as one that has the following properties:

[Definition: schema document set coherence]

> A schema document set is coherent when it has the following properties: (1) the set does not refer to a schema document outside the set (i.e., the set is closed), and (2) the set does not include two different versions of the same component in an incompatible way.

Consider the following simple example of incoherence in the figures below. Consider Figure 2-1, in which Justice domain has published a new schema document (version 4.1). Note the descendant relationships between the old and new data components. A schema document set consisting of Screening 1.1 and Justice 4.1 is incoherent because it refers to the old Justice 4.0 schema document outside the set, and therefore, violates the first criterion (the set must be closed). To resolve this we could incorporate the older 4.0 version into this set. Figure 2-2 indicates that adding Justice 4.0 violates the second criterion because multiple versions of the same component will exist that are incompatible. To make a coherent schema document set, either the Screening domain must be adjusted to use the new Justice 4.1 component or the schema document set must be revised to use the Screening domain with Justice 4.0 and not Justice 4.1.

Figure 2-1. Incoherent schema set - not closed.

Figure 2-2. Incoherent schema set - incompatible data components.

In general, two or more versions of a data component are incompatible when a type or element in one version of a schema has been copied to or redefined/redeclared in another, and both versions must exist in the same set because of cross referencing (as in the figure above). Note that even if all data components have not changed within two versions of the same schema, a set that contains both schema documents will still be incoherent because the mere duplication of a data component in a new namespace is considered redefinition (and, of course, duplication).

However, two versions of a data component can also exist in a compatible way. The compatibility of two different versions of a data component depends on the way the ancestor component was changed to obtain the descendant. In Figure 2-2, Justice 4.1 and 4.0 Arrest elements are incompatible because the 4.1 version of Arrest was simply given an additional property (NewElement) and is essentially a redeclaration of the 4.0 version. This results in two semantically identical elements. In fact, as already mentioned, even if the ArrestType had remained the same across both versions, the 4.1 version is considered a redefinition and duplication of the 4.0 version.

On the other hand, if the 4.1 ArrestType had been derived (through type derivation) from the 4.0 version, and the 4.1 Arrest element had been made substitutable for the 4.0 version, then these components would be compatible. The difference is that these components have a clear relationship to their ancestors that is defined through XML mechanisms, whereas the former components do not. Furthermore, the substitutability property makes these components easily usable together (i.e., compatible).

The need to be a coherent schema document set is ONLY required by official NIEM releases (major, minor, and micro). A core update is not absolutely required to be coherent with the core it applies to. However, except in rare cases, it will be crafted to be coherent. In order to provide flexibility to domains, a domain update schema document set is not required to be coherent. Whether or not a domain update is coherent with a given release, is dependent upon its change log which indicates how it changes the schema documents it applies to.

2.9. MPD Types

This section details the five classes of MPDs currently defined in NIEM.

2.9.1. NIEM Release

A NIEM release is an MPD containing a full set of harmonized reference schema documents that coherently define and declare all content within a single version of NIEM. NIEM releases include major, minor, and micro releases (as defined in the NIEM High Level Version Architecture (HLVA) **[NIEM High-Level Version Architecture]**).

[Definition: release]

A reference schema document set published by the NIEM Program Management Office (PMO) at http://release.niem.gov/ and assigned a unique version number. Each schema defines data components for use in NIEM information exchanges. Each release is independent of other releases, although a schema document may occur in multiple releases. A release is of high quality, and has been vetted by NIEM governance bodies. A numbered release may be a major, minor, or micro release.

Current real examples of NIEM releases include NIEM major releases 1.0, 2.0, and 3.0, and minor release 2.1. Each numbered release is a reference schema document set that includes a NIEM Core (along with the various infrastructure and code list schema documents that supplement Core) and NIEM domain schema documents.

[Definition: major release]

A NIEM release in which the NIEM Core reference schema document has changed since previous releases. The first integer of the version number indicates the major release series; for example, versions 1.0, 2.0, and 3.0 are different major releases.

[Definition: minor release]

A NIEM release in which the NIEM Core has not changed from previous releases in the series, but at least one or more domain reference schema documents have changed. A second digit greater than zero in the version number indicates a minor release (for example, v2.1). Note also that major v2.0 and minor v2.1 are in the same series (i.e., series 2) and contain the same NIEM Core schema document.

[Definition: micro release]

A NIEM release in which neither the NIEM Core nor the domain reference schema documents have changed from the previous major or minor release, but one or more new reference schema documents have been added (without impact to domain or Core schemas). A third digit greater than zero in the version number indicates a micro release (for example, v2.1.1 note that this release does not exist as of this date).

A micro release is a NIEM release that adds new data components to the Core, domains, or both without removing or modifying existing Core and domain schemas or content. [Figure 2-3] illustrates both real (v1.0, v2.0, v2.1, and v3.0) and fictitious (v2.1.1 and v2.1.2) examples of major, minor, and micro release composition.

Note that a given NIEM reference schema document (target namespace) can exist in multiple numbered releases. For example, as illustrated in Figure 2-3, both NIEM 2.0 and 2.1 contain (reuse) the same NIEM Core 2.0 schema document. Reuse of schema documents among releases is carefully coordinated to ensure coherence is maintained within each release. The **[NIEM High-Level Version Architecture]** defines the processes for numbering releases and identifying the schema documents that compose these sets. Later, this specification will outline a similar version numbering scheme for MPDs and their artifacts.

Figure 2-3. Examples of NIEM numbered releases.

2.9.2. Domain Update

A *domain update (DU)* is an MPD containing reference schema document or document set that represents changes to NIEM domains. The **[NIEM High-Level Version Architecture]** defines a domain update as both a process and a NIEM product. Through use and analysis of NIEM releases and published content, domain users will identify issues and new data requirements for the domain and sometimes Core. NIEM domains use these issues as the basis for incremental improvements, extensions, and proposed changes to future NIEM releases. Both the process and product of the process are referred to as domain update. This MPD Specification is applicable to a domain update product.

[Definition: domain update]

A MPD that contains a reference schema document or document set issued by one or more domains that constitutes new content or an update to content that was previously included in a NIEM release. A domain update may define and declare new versions of content for NIEM releases or other published content. The issuing body vets each update before publishing, but the update is not subject to review by other NIEM bodies. A domain update must be NIEM-conformant, but otherwise it has fewer constraints on quality than does a NIEM release. Domain update schema documents contain proposed future changes to NIEM that have not been published in a numbered release and have not been vetted by

NIEM governance bodies (except by the domain or domains involved). Domain updates are published to the NIEM Publication Area at `http://publication.niem.gov/` and available for immediate use within IEPDs.

A *domain update* represents changes to domain schemas. The primary artifacts that define these changes are one or more reference schema documents and a change log. A domain update may apply to one or more domain namespaces within a single NIEM major, minor, or micro release. A domain steward uses a domain update to: (1) make new or changed domain content immediately available to NIEM data exchange developers between NIEM releases, and (2) request that new or changed content be harmonized into a future NIEM release. (See the *Domain Update Specification* **[NIEM Domain Update Specification]** which provides normative details about domain updates and the associated processes.)

2.9.3. Core Update

When necessary, the NIEM PMO can publish a *core update (CU)*. This is essentially identical to a domain update in terms of structure and use, with two important exceptions. First, a core update records changes that apply to a particular NIEM core version or another core update. This also means it is applicable to all NIEM releases using that same core version. Second, a core update is never published to replace a NIEM core. It is intended to add new schemas, new data components, new code values, etc. to a core without waiting for the next major release. In some cases, minor modifications to existing data components are possible.

[Definition: core update]

An MPD that applies changes to a given NIEM core schema document or document set. A core update never replaces a NIEM core; instead, it is used to add new schema documents, new data components, new code values, etc. to a particular NIEM core. In some cases, a core update can make minor modifications to existing core data components.

As with domain updates, all core updates are published to the NIEM Publications Area, their changes are immediately available for use in IEPDs, and they will be harmonized and integrated into the next major NIEM release.

2.9.4. Information Exchange Package Documentation (IEPD)

NIEM Information Exchange Package Documentation (IEPD) is an MPD that defines a class of instance XML documents that represent a recurring XML data exchange.

[Definition: Information Exchange Package Documentation (IEPD)]

An MPD that defines one or more (generally recurring) XML data exchanges.

A NIEM IEPD is a NIEM-conformant XML schema document set that may include portions of a NIEM Core schema document (and updates), portions of NIEM Domain schema documents (and updates), enterprise-specific or IEPD-specific extension schema documents, and at least one XML document element (as defined in **[W3-XML-InfoSet]**) declared in a schema document. The XML schema documents contained in an IEPD work together to define a class of instance XML documents that consistently encapsulate data for meaningful information exchanges. Each instance XML document in this class validates against the XML schema document set contained in the IEPD. XML schema documents in a NIEM IEPD conform to the **[NIEM NDR]** and may use or extend data component definitions drawn from NIEM.

An IEPD consists of a minimal but complete set of artifacts (XML schema documents, documentation, sample instance XML documents, etc.) that together define and describe an implementable NIEM information exchange. A complete and normative IEPD should contain an XML schema document set and instructional material necessary to:

- Understand information exchange context, content, semantics, and structure.
- Create and validate information exchanges defined by the IEPD.
- Identify the lineage of the IEPD and optionally its artifacts.

An NIEM IEPD defines an Information Exchange Package (IEP), which is an instance XML document that validates to the schema document set in the IEPD. An IEP is an information message payload as it will appear when transmitted electronically and serialized as XML. (**[FEA DRM]** and **[GJXDM IEPD Guidelines]** and are the original sources of the terms information exchange package and information exchange package documentation, respectively).

[Definition: Information Exchange Package (IEP)]

An XML document that is a valid instantiation of a NIEM IEPD, and therefore, validates with the schema document set of that IEPD.

It is NOT a requirement of NIEM conformance that an IEP be native XML on the transmission medium. A NIEM IEP that is encrypted, compressed (e.g., using Zip, RAR, **[W3-EXI]**, etc.), or wrapped within an envelope mechanism is still considered NIEM-conformant if its original native form can be extracted by the receiver.

2.9.5. Enterprise Information Exchange Model (EIEM)

As an organization develops IEPDs, the organization may realize that many of its IEPDs have similar business content. A collection of closely related business data could be organized at an object level and defined as extension data components. In NIEM, these extension components are referred to as *Business Information Exchange Components (BIECs)*, because they are either specific to an organization's business or they represent a more general line of business that crosses organizational lines. Often they are business data components developed and used by multiple organizations within the same community of interest. So, instead of an "organization," it is more appropriate and provides better context if we use the term *information sharing enterprise*.

[Definition: Information Sharing Enterprise]

A group of organizations with business interactions that agree to exchange information, often using multiple types of information exchanges. The member organizations have similar business definitions for objects used in an information exchange and can usually agree on their common BIEC names and definitions.

Information sharing enterprises may cross various levels of government and involve multiple business domains. They are self-defining and can be formal (with specific governance) or informal and *ad hoc*. An information sharing enterprise is the primary entity that supports the development and management of BIECs and an associated Enterprise Information Exchange Model (EIEM) (to be discussed next). Henceforth, unless otherwise stated, all references to an enterprise will implicitly mean information sharing enterprise.

A *Business Information Exchange Component (BIEC)* **[NIEM BIEC]** is a NIEM-conformant content model in XML Schema for a data component that meets the specific business needs of an information sharing enterprise for exchanging data about something that is a part of one or more information exchanges. This data component is tailored and intended to be used consistently across multiple IEPDs built by an enterprise. A BIEC is a NIEM-conformant data component that is:

- Reused from a NIEM release (for example, as a subset; with possibly modified cardinality), or
- Extended per the **[NIEM NDR]** from an existing NIEM data component, or
- Created per the **[NIEM NDR]** as a new data component that does not duplicate existing NIEM components within a release in use.

[Definition: Business Information Exchange Component (BIEC)]

A NIEM-conformant XML schema data component definition or declaration (for a type, element, attribute, or other XML construct) reused, subsetted, extended, and/or created from NIEM that meets a particular recurring business requirement for an information sharing enterprise.

The use of BIECs has the potential for simplifying IEPD development and increasing consistency of the business object definitions at all steps in the process, including exchange content modeling, mapping to NIEM, creating NIEM extension components, and generating XML schema documents.

An *Enterprise Information Exchange Model (EIEM)* is an MPD that incorporates BIECs that meet enterprise business needs for exchanging data using NIEM **[NIEM BIEC]**. An EIEM is an adaptation of NIEM schema documents, tailored and constrained for and by an enterprise. An EIEM contains the following schema documents that are commonly used or expected to be used by the authoring enterprise:

- One standard NIEM subset schema document set (or reference schema set). One or more NIEM extension schema documents that extend existing NIEM data components or establish new NIEM-conformant data components.

- Optionally, as needed, one or more NIEM constraint schema document sets (usually based on a subset schema document set).
- Optionally, as needed, one or more XML schema documents for non-NIEM (i.e., non-conformant) standards with associated extension schema documents that contain adapter types for the data components that will be used from those non-NIEM XML schema documents (per **[NIEM NDR]**).

[Definition: Enterprise Information Exchange Model (EIEM)]

> An MPD that contains a NIEM-conformant schema document set that defines and declares data components to be consistently reused in the IEPDs of an enterprise. An EIEM is a collection of BIECs organized into a schema document subset and one or more extension schema documents. Constraint schema documents and non-NIEM-conformant external standards schema documents with type adapters are optional in an EIEM.

An information sharing enterprise that creates and maintains an EIEM authors IEPDs by reusing its EIEM content instead of (re)subsetting reference schema documents sets and (re)creating extensions. An EIEM may also contain business rules or constraint schema document sets tailored to the enterprise and designed to restrict variability in use of NIEM data components. This not only saves time, but it also ensures that enterprise IEPDs reuse NIEM and associated extensions consistently. (Note that XML schema document subsets, extension schema documents, and constraint schema document sets will be defined and discussed in more detail later in this document). [Figure 2-4] below illustrates relationships among BIECs, an EIEM, and an IEPD family (Constraint schema document sets are optional and not depicted in [Figure 2-4]).

Figure 2-1: BIECs, EIEM, and a small family of IEPDs.

```
<image src="img/iepd-family.png"/>
```

2.10. Similarities and Differences of MPD Classes

It will be helpful to summarize the foregoing discussions by listing the primary similarities and differences among the various types of MPDs. This will help highlight the nature of this specification as a baseline and point of leverage for all five classes of MPDs: NIEM release, core update, domain update, IEPD, and EIEM. Note that these lists are not all inclusive.

MPD class similarities:

- Principal artifacts are XML schema documents (XSD), the purpose for which is to define and declare reusable data components for information exchanges or to define the exchanges themselves.
- Each MPD requires a self-documenting mpd-catalog.xml artifact (containing metadata and a listing of all its artifacts), which establishes its purpose, lineage, organization, etc.

- Each MPD requires a change log.
- Each MPD requires a Uniform Resource Identifier (URI) and a version number.
- Each MPD must be packaged as a self-contained ZIP archive (in one form). Self-contained simply means that an MPD has copies of (not just URLs or references to) all the schemas needed to validate instances it defines.
- Each MPD may contain optional alternate representations besides XML Schema (for example, a generic graph, a UML graph, XMI, database format, spreadsheet, etc.).

MPD class differences:

- IEPDs and EIEMs contain subset, extension, and constraint schema documents and document sets. NIEM releases, core updates, and domain updates contain reference schema document sets.
- An IEPD must declare at least one XML document element in an XML exchange schema document. Other MPD classes do not have this requirement.
- An IEPD must contain at least one sample instance XML document corresponding to each document element declared in exchange schemas. An XSLT artifact to display instances is optional.
- EIEMs and domain updates may optionally contain sample instance XML documents and associated XSLT files to display them. NIEM releases and core updates do not.
- Domain updates supersede other published schema documents/namespaces and do not include other unchanged content. IEPDs, EIEMs, and NIEM releases are independently complete. Core updates can only supplement and never replace a NIEM core. However, a core update can be issued as a new complete standalone reference schema document to be used with a NIEM core.

The following table summarizes the similarities and differences of MPD classes by indicating the characteristics for each:

Figure 2-2: Comparison of MPD classes.

```
<image src="img/compare-mpd.png"/>
```

Table 2-1: Table 1. Comparison of MPD classes.

Characteristics of MPD Classes	Release	CU	DU	IEPD	EIEM
Requires a URI	X	X	X	X	X
Requires a version number	X	X	X	X	X
Must be packaged as a ZIP archive	X	X	X	X	X
May contain alternate model representations (in addition to XSD)	X	X	X	X	X
Requires self-documenting XML mpd-catalog (specified by XSD)	X	X	X	X	X
Requires a formal XML change log (specified by XSD)	X	X	X		
Requires a change log but may be informal; any format				X	X

Requires a master document				X	X
Its XML schema document set defines reusable data components	X	X	X		X
Its XML schema document set defines data exchanges (IEPs)				X	
Can contain subset, extension, constraint, and/or exchange schema documents				X	
Contains subset and extension schema documents; optionally constraint schema document sets					X
Contains reference schema documents only	X	X	X		
Must declare at least one or more XML document elements; thus, requires at least one XML exchange schema document				X	
May contain sample instance XML documents that validate to XML schema document set			X	X	X
Required to be independently complete standalone XML schema document set	X			X	X
May be independently standalone XML schema document set		X	X		
May supersede other published XML schema documents (target namespaces>			X		

3. MPD XML Schema Artifacts

XML schema document artifacts are the essential content of MPDs because they normatively define and declare data components. The purpose of an MPD is determined by the XML schema document or document set(s) it contains; furthermore, each schema document may have a different purpose. The **[NIEM NDR]** addresses various types of schema documents as conformance targets: reference, extension, and constraint schema documents. Each conformance target may adhere to a different (though possibly overlapping) set of conformance rules. Consult the **[NIEM NDR]** for these rules.

Note that exchange schema document is not a conformance target. The same set of NDR rules for extension schema documents applies to both. This is because in an IEPD an extension schema document that declares one or more top-level XML document (sometimes referred to as root) elements is also identified as an exchange schema document.

The following subsections will define each type of NIEM schema and identify the types of MPDs that may or must contain them. The last subsection discusses sample instance XML documents (IEPs) that validate with IEPD schema document sets, and when such instance XML documents are mandatory.

3.1. Reference Schemas

This section generally applies to NIEM releases, core updates, and domain updates. Though not common, it is also valid to use a reference schema document or document set within an IEPD or EIEM. Reference schema document and reference schema document set were defined earlier in Section 2.7, *Reference Schema Document*, above.

A NIEM reference schema document is intended to be the authoritative definition schema document for a NIEM target namespace, therefore, all NIEM releases, core updates, and domain updates are composed of a reference schema document set and associated namespaces. As a standalone artifact, a reference schema document set is always coherent and harmonized such that all types and properties are semantically unique (i.e., multiple versions of semantically identical types or properties do not exist within the set).

As authoritative definitions, NIEM reference schema document sets satisfy more rigorous documentation requirements. The **[NIEM NDR]** requires that each type definition, and element and attribute declaration in a reference schema document contain an `xs:annotation` element that defines its semantic meaning. As will be explained later, extension schema documents are also authoritative definitions, but in a local sense. They are authoritative within a given IEPD or EIEM, and therefore, must also satisfy the same rigorous documentation rules as reference schema documents.

Typically reference schema documents contain data components with the most relaxed cardinality (zero to unbounded repetition). However, this is not an absolute requirement. Cardinality in reference schema documents may be constrained if necessary to model reality. For example, we might say that NIEM releases should restrict a Person object to a single occurrence of BirthDate. Unfortunately, also in reality, criminal persons often present multiple identities and multiple birth dates; and so the capability to represent such is an important data requirement of NIEM.

3.2. Subset Schemas

This section only applies to IEPDs and EIEMs. NIEM releases, core updates, and domain updates do not contain schema document subsets (only reference schema document sets).

3.2.1. Basic Subset Concepts

A NIEM XML schema document subset is a set of subset schema documents that constitutes a reduced set of components derived from a NIEM reference schema document or document set associated with a given numbered release or domain update. Any given XML schema document within a schema document subset is referred to as a subset schema document (terms reversed). The primary purpose for a schema document subset is to reduce and constrain the scope and size of a full NIEM reference schema document set for use within an IEPD or EIEM.

[Definition: subset schema document]

An XML schema document that meets all of the following criteria:

- It is built from a referenc schema document set where one or more reference schema documents has been substituted by a its corresponding subset schema document.
- It is built from a reference schema document by applying subset operations to the XML schema statements in a reference schema document.
- It is explicitly designated as a subset schema document. This may be declared by an MPD catalog or by a tool-specific mechanism outside the subset schema document.
- It has a target namespace previously defined by a reference schema document. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema document.

- It does not alter the business semantics of components in its namespace. The reference schema document defines these business semantics.
- It is intended to express the limited vocabulary necessary for an IEPD or EIEM and to support XML Schema validation for an IEPD.

See also **schema document subset**.

Schema document subsets have been derived from a related reference schema document set (such as a NIEM release).

[Definition: schema document subset]

An XML schema document set built from a reference schema document set by applying subset operations to that reference schema documents in that set. See also **subset schema document**.

Because NIEM adopts an optional and over-inclusive data representation strategy, most elements in a NIEM reference schema have zero to unbounded cardinality. So, elements with cardinality `minOccurs="0"` are optional and may be omitted from a subset schema document if not needed for business reasons. It is also valid to constrain element cardinality within a subset schema document, as long as it is not constrained such that the subset relationship is broken. For example, a reference schema document element with cardinality (`minOccurs="0"`, `maxOccurs="unbounded"`) may be constrained to (`0`,`1`) or (`1`,`1`) in a subset schema document. However, if a reference schema document element's cardinality is (`1`,`unbounded`), it may not be constrained to (`0`,`1`) since this breaks the subset relationship. The interval (`0`,`1`) is not contained within, and instead, overlaps the interval (`1`,`unbounded`).

The fundamental rule for a valid schema document subset is as follows:

[Rule 3-1]

31 Any instance XML document that validates against a NIEM schema document subset will validate against the NIEM reference schema document set from which that schema document subset was derived.

NIEM Subset Operations: These are essentially reduction operations that remove or constrain portions of a reference schema document set, thereby building a profile of the set. They do not expand the scope (i.e., relax constraints) or change the semantics of reference schema document set content.

1. Remove an XML comment statement
2. Remove an `xs:annotation` (includes `xs:documentation` and `xs:appinfo`)
3. Increase the value of an `xs:element minOccurs` attribute (must be less than or equal to

`maxOccurs` value)

4. Decrease the value of an `xs:element maxOccurs` attribute (must be greater than or equal to `minOccurs` value)
5. Remove an `xs:element` if `minOccurs="0"`
6. Remove an `xs:complexType` or `xs:simpleType` (if not supporting an element)
7. Remove an `xs:attribute` from a `xs:element`
8. Change an `xs:attribute use="optional"` to `use="prohibited"`
9. Change an `xs:attribute use="optional"` to `use="required"`
10. Remove an `xs:schema/xs:element` declaration (if not supporting an element use)
11. Set an `xs:schema/xs:element` to `abstract="true"`
12. Substitute a `substitutionGroup` member `xs:element` for its associated substitution head
13. Remove an `xs:enumeration` from an `xs:simpleType` (unless it is the only remaining `xs:enumeration`)
14. Add or apply a constraining facet to an `xs:simpleType`
15. Remove an `xs:import` and its associated schema document (if the schema document is not used within the document set)

Note that the process of deriving a schema document subset from a NIEM reference schema document set is optional; it is valid to reuse NIEM reference schema documents as-is within IEPDs or EIEMs. The primary reasons for subsetting are to reduce IEPD size and complexity and to focus constraints.

3.2.2. Subset Namespaces

A subset is essentially a reference schema set (numbered release) that has been modified per the above rules to support business requirements represented in an IEPD or EIEM. A subset derived from a reference schema set may differ from that reference set only in that its content has been reduced and/or constrained. For this reason, schemas in a subset adopt target namespaces that are identical to the corresponding schemas in the reference schema set.

[Rule 3-2]

34 Each subset schema document in a schema document subset derived from a reference schema document set bears the same target namespace as the schema in the reference schema document set on which it is based.

3.2.3. Omitting Schemas in Subsets

In some cases, an entire schema appearing in a reference set may be omitted from a corresponding subset. The following rule specifies when this is valid.

[Rule 3-3]

35 A schema document contained in a reference schema document set may be omitted from a derived schema document subset, if and only if no elements, attributes, or types declared or defined in the schema document are required to support other elements or types within the schema document subset for exchange purpose.

When a schema document is not required, and therefore, has been omitted from a valid schema document subset, there is no longer a reason to import it from anywhere within the subset. Therefore, because some XML validators (and other tools) expect the file identified in the `xs:import/schemaLocation` attribute to be present, it's a good idea to remove such references to the omitted schema.

3.2.4. Multiple Subsets in a Single IEPD or EIEM

This section only applies to NIEM IEPDs and EIEMs. NIEM releases, core updates, and domain updates do not contain schema subsets.

Previous sections defined a single schema subset derived from a reference schema set. In general, an IEPD or EIEM contains a single cohesive schema subset (which may be a rather large set of files) based on one numbered NIEM release or domain update.

However, this specification does not restrict the number of different subsets that may be employed within a single IEPD or EIEM. Furthermore, it does not restrict the employment of subsets from different numbered releases within a single IEPD or EIEM. However, exercising this degree of flexibility makes it critically important that developers understand the potential consequences. NIEM subsets represent a delicate compromise between flexibility and interoperability. On the one hand, a set of IEPDs based on the same subset and numbered release use identical data components, thereby enhancing interoperability. On the other hand, mixing dissimilar subsets from the same numbered release or mixing subsets derived from various numbered releases has the potential to negatively impact interoperability.

The NIEM mandate that every schema have a unique namespace prevents name conflicts between reference schema sets and between two subsets derived from different reference sets. In spite of namespace distinction, mixing subsets of multiple reference schema sets can still introduce multiple versions of semantically equivalent data components, a potentially ambiguous situation. Even employing multiple subsets together that have been derived from the same reference set has the potential to create a similar result. Above all, it is the developer's responsibility to ensure that, if mixing subsets from one or more numbered releases within a single IEPD or EIEM, these artifacts are carefully coordinated and clearly documented to ensure the various versions of semantically equivalent data components and different schemas with the same namespaces will not cause conflicts, confusion, and/or failure during validation or exchange implementation.

3.3. Extension Schemas

This section only applies to NIEM IEPDs and EIEMs. NIEM releases, core updates, and domain updates do not contain extension schemas.

[Definition: extension schema document]

> A NIEM-conformant schema document that adds domain or application specific content to the base NIEM model.

The normative definition for a NIEM IEPD extension schema is in the **[NIEM NDR]**. In general, an extension schema contains components that use or are derived from the components in reference schemas. It is intended to express the additional vocabulary required for an IEPD, above and beyond the vocabulary available from reference schemas.

A IEPD or EIEM developer who determines that NIEM does not contain all elements required for a given information exchange has two options to account for such requirement shortfalls. Using rules and techniques outlined in the **[NIEM NDR]**:

- Extend an existing NIEM data component.
- Build a new NIEM-conformant data component.

A NIEM extension schema may contain data components built from both options above. Employment of extension schemas in an IEPD is entirely optional.

Multiple extension schemas are allowed in a single IEPD. Developers will likely want to reuse many of their extensions in other IEPDs. Therefore, it is most efficient to put all extensions designed for reuse into one extension schema (or into a well-organized set of extension schemas), while keeping IEPD-specific extensions in a different schema. The reusable extension schema or schema set can then be easily redeployed in other IEPDs as needed.

Extension schemas generally contain new data component declarations that may (though not necessarily) be derived from or reference existing NIEM components. This being the case, reference schemas do not exist for new data components found within extension schemas. Therefore, extension schemas must satisfy the more rigorous documentation requirements of reference schemas in accordance with the **[NIEM NDR]**. The definition or declaration of each new data component (type, element, or attribute) in an extension schema includes an `xs:annotation` element that provides its semantics and NIEM-specific relationships.

3.4. Exchange Schemas

This section only applies to IEPDs. NIEM releases, core updates, domain updates, and EIEMs do not contain exchange schemas.

An IEPD defines one or more NIEM XML data exchanges, and therefore, a class of instance XML documents, each of which validates against the XML schema document set in that IEPD. An instance XML document contains exactly one document element, which is the root element of the instance, and which cannot appear in the content of any other element within that instance XML document (by definition in **[W3-XML]** and **[W3-XML-InfoSet]**.

An IEPD exchange schema declares one or more document elements. Only one document element can be used in any given information exchange instance (IEP).

[Definition: exchange schema document]

> A NIEM-conformant schema document that declares an XML document element (top-level) for a particular information exchange.

However, a NIEM IEP (instance) could later become the payload of an XML envelope (such as a SOAP message). That XML envelope (itself an XML document instance) will have its own document element. In this case, the IEP no longer contains the document element for the instance. Therefore, in the context of an IEPD, it is more appropriate to refer to a document element as a root element of an IEP.

[Definition: IEP root element]

> The single top-level element in an IEP (instance XML document). In the absence of any other XML wrapping of an IEP, a root element declared in an exchange schema document is an IEP document element.

This results in the following rule:

[Rule 3-4]

36 An IEPD MUST contain at least one exchange schema artifact that declares at least one top-level root element for IEP instances specified by the IEPD.

Note that neither **[W3-XML]** nor this rule restricts the number of root elements that can be declared by an IEPD exchange schema. Any XML schema document, including a NIEM exchange schema, may declare multiple root elements. Furthermore, a single IEPD may contain multiple exchange schemas if such are necessary to meet business requirements.

Nonetheless, regardless of the number of root elements declared by a given exchange schema, and in accordance with **[W3-XML]**, any XML instance (i.e., IEP) that validates against a NIEM IEPD must contain one and only one root element and that element must be declared within at least one exchange schema in the IEPD.

Similar to the case of multiple subsets, the flexibility provided by allowing multiple root elements and multiple exchange schemas in a single IEPD has the potential to be both powerful and problematic. Again, developers are responsible to carefully coordinate and clearly document multiple roots and/or multiple exchange schemas in a single IEPD to prevent ambiguity and misinterpretation related to validation, implementation, and use. (See Section 5 Optional MPD Artifacts for documentation artifacts that an IEPD may include when needed.)

The **[NIEM NDR]** does not allow locally declared elements; all NIEM elements are declared with global scope at the top level. This means that any new element declaration in a NIEM-conformant exchange schema has the potential to be the root element in a corresponding IEP. Therefore, if an IEPD author does not intend for a new element to be an IEP root element, then do not declare it in a NIEM exchange schema. Declare it in an extension schema instead.

[Rule 3-5]

37 An IEPD exchange schema MUST NOT declare any XML element that is not intended for use as an IEP root element.

Note that this rule does not preclude the use (through `ref=`) of other elements within the exchange schema that are declared globally elsewhere within an IEPD. In general, elements that must be used within an exchange schema, but are not intended to be IEP root elements should be declared in extension schemas.

Now that both extension and exchange schemas have been described, it is useful to mention the following special case. Although generally rare, it is possible to develop an IEPD without an extension schema. If an author creates an IEPD based entirely on existing NIEM elements, then no extension schema is necessary (and this is the reason it is deemed optional). In this case, the exchange schema defines a root element type to contain only existing NIEM elements (i.e., drawn from a subset or reference schemas) and declares the root element of that type. The rule above merely ensures that the exchange schema declares root element(s) that are intended and will be used as the roots of XML instances, and no others. However, because all NIEM elements must be declared with global scope **[NIEM NDR]**, in order to declare non-root elements in an IEPD, they will have to be declared in an extension schema.

It is usually good practice to maintain namespace cohesion. If root elements declared in an IEPD tend to change together, then group them together in the same namespace (i.e., single exchange schema). If elements tend to change independently, then group them in separate namespaces (i.e., multiple exchange schemas). Furthermore, while there are no restrictions on the number of root elements and exchange schemas, it may be best to first consider declaring one root per exchange schema if this arrangement can support the IEPD business requirements. If not, only then consider scaling upward. In all cases, clearly document the intent and rationale for how an IEPD is organized and to be implemented.

3.5. Base Schema Set

Within an MPD, the base schema document set is the XML schema document set that defines the information exchange or model in the MPD. This set may incorporate NIEM-conformant reference, subset, extension, and exchange schema documents, as well as schema documents from an external non-NIEM-conformant source (for example, GML, Geospatial Markup Language).

[Definition: base schema document set]

A NIEM MPD artifact that is the set of all NIEM-conformant and external non-conformant XML schema documents that together specify an information exchange or information model for an MPD. A base schema document set may incorporate reference, subset, extension, and exchange schema documents, as well as external schema documents from authoritative sources outside of NIEM. (Constraint schema documents and sets are NOT part of a base schema document set.)

An MPD contains one and only one base schema set.

[Rule 3-6]

38 An MPD MUST contain one and only one base schema set.

3.6. Constraint Schemas

This section only applies to NIEM IEPDs and EIEMs which may use constraint schemas. NIEM releases, core updates, and domain updates do not contain constraint schemas.

A constraint schema is an optional IEPD or EIEM artifact that is used to express business rules for a class of XML documents, and is not assumed to be a definition for the semantics of the components it contains and describes. Instead, a constraint schema uses the XML Schema definition language to add constraints and restrictions to components defined or declared by other schemas, usually from a schema subset.

[Definition: constraint schema document]

A schema document which imposes additional constraints on NIEM-conformant instances. A constraint schema document or a constraint schema document set validates additional constraints imposed on an XML instance only after it is known to be NIEM-conformant (i.e., has been validated to reference schema documents, subset schema documents, extension schema documents, and/or exchange schema documents). Constraint schema validation is a second-pass validation that occurs independently of and after conformance validation. A constraint schema need not validate constraints that are applied by other schemas. See also **constraint schema document set**.

[Definition: constraint schema document set]

A set of related constraint schema documents that work together, such as a constraint schema document set built by adding constraints to a schema document subset. See also **constraint schema document**.

Note that constraint schemas are generally useful when it is necessary to impose restrictions that are more complex than cardinality. If only cardinality restrictions are needed, then it is easier and more efficient to set these directly in the schema subset and avoid the use of constraint schemas. Otherwise, a constraint schema may be necessary. Note however, that any cardinality restrictions placed on NIEM release components within subsets must not violate the rules established in 3.2.1 Basic Subset Concepts which define the relationship of a subset schema to the reference schema on which it is based.

The **[NIEM NDR]** provides a normative definition and description of constraint schemas. However, a few points are worth mentioning here.

Use of constraint schemas is one option for applying additional business rules to or tightening constraints on NIEM IEPs beyond what NIEM itself provides. This particular technique uses XML Schema. NIEM also allows other methods that do not use XML Schema, such as **[ISO Schematron]** or other language methods. However, at this time there are no normative rules for how these techniques should be employed in NIEM IEPDs or EIEMs. Therefore, if other techniques are used, it is a developer responsibility to incorporate appropriate artifacts and clear documentation.

Constraint schemas are generally designed and employed in sets, similar to schema subsets and reference sets. A common practice for creating an IEPD or EIEM constraint schema set is to start with a valid NIEM subset and modify this set to further restrict the class of XML documents (IEPs) that will validate with this constraint set. However, an extension or exchange schema can also be used to derive a constraint schema. The namespace of a constraint schema is established the same way the namespace of a subset schema is established it reuses the target namespace of the schema from which it is derived.

[Rule 3-7]

39 A constraint schema bears a target namespace that has been previously assigned by a reference schema, extension schema, or exchange schema, or is a schema that is intended to support a constraint schema that has such a target namespace.

To use a constraint schema to tighten constraints on IEPs, a two-pass validation technique is employed. In the first pass, an IEP is validated against the schema subset, extensions, and one exchange schema. This pass ensures that IEP semantics and structure conform to the NIEM model and NDR. In the second pass, an IEP is checked against a constraint schema set, which may contain constrained versions of the schema subset, extensions, and the appropriate exchange schema. This pass ensures that the IEP also satisfies the additional constraints (i.e., business rules that the first pass was unable to validate).

There is no restriction on the number of constraint schema sets that an IEPD or EIEM can employ. As in other advanced situations, developers must clearly document their intentions for and use of multiple constraint schema sets.

In general, constraint schemas have far fewer requirements than other classes of NIEM schemas. Since they work in tandem with NIEM normative schemas, constraint schemas are allowed to use the XML Schema language in any way necessary to express business rules. This means that to constrain instances, constraint schemas can employ XML Schema constructs that are not allowed in other classes of NIEM schemas.

BIECs in particular may have additional business rules in constraint schemas. A normative NIEM BIEC Specification (in progress at the time of the publication of this MPD Specification), will supplement or obviate constraint schemas with consistent and formal techniques for representing business rules within NIEM components. However, as already mentioned, the MPD Specification does not prohibit or restrict the application of formal business rule techniques to MPDs now.

Finally, within an MPD, constraint schemas and constraint schema sets are completely distinct from the base schema set. In the future, NIEM will prefer business rules over constraint schemas as the primary method for further constraining a base schema set.

3.7. Classes of MPDs vs. Classes of Schemas

The chart in Table 3-1 summarizes which types of schemas are contained in which classes of MPDs and where they are not applicable (NA = Not Applicable; U = unbounded).

Notice that only NIEM releases, core updates, and domain updates contain reference schema sets, while only IEPDs and EIEMs contain the user developed schema sets. Since an IEPD defines at least one data exchange, it must contain at least one exchange schema. Furthermore, the diamonds (F) indicate that a NIEM-conformant IEPD or EIEM must have at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema (See **[Rule 3-8]** below this table).

Table 3-1. MPD classes vs. schema classes

[Rule 3-8]

310 A NIEM-conformant IEPD or EIEM MUST contain at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

3.8. Sample XML Instances

XML schema document sets define data exchange instance XML documents. It is certainly possible to construct an example instance XML document for an entire NIEM release, but such an instance is of questionable value. Rarely, if ever, will an entire NIEM release be used to define a data exchange. However, sample instance XML documents are very valuable artifacts to include with IEPDs. As samples of the actual exchange data, instance XML documents can help an IEPD implementer to understand the original intent of the IEPD developer. They can be used by an implementer as data points for validation with the IEPD base schema document set or a constraint schema document set. And finally, the inclusion of valid, meaningful sample instance XML documents is an indication of IEPD quality. For these reasons, IEPDs require valid sample instance XML documents.

[Rule 3-9]

311 A NIEM IEPD MUST contain at least one valid sample instance XML document (i.e., IEP) artifact for each exchange schema element that can be the root of a corresponding IEP.

The intent of this rule is not to provide a test for all permutations of XML instances that might be possible given the schema definitions and declarations. As the value propositions explain, its purpose is to ensure IEPD developers have tested their own designs, and to provide implementers with examples they can use for additional understanding and guidance. IEPD developers should strive to include sample XML instances that (1) capture real world business cases of data exchanges, and (2) exercise as many data component definitions and declarations in the schemas as possible. While both of these goals may not be achievable in a single sample XML instance, developers have the option to include multiple sample XML instances; however, only one per intended root element is mandatory.

Note that each sample XML instance illustrates one view of the data based on a chosen set of conditions that apply to an IEP. Other views based on different conditions likely exist. A developer must review the business rules and other documentation in an IEPD to ensure understanding of all possible conditions. Do not rely exclusively on sample XML instances, since they are not required to account for all IEP permutations.

This specification encourages but does not mandate the inclusion of sample XML instances for EIEMs and domain updates. Again, such instances may be valuable to user understanding of the intent and usage of data components (especially, those that are new, extended, or changed).

4. MPD Documentation Artifacts

A variety of documentation files may be incorporated into a NIEM MPD. However, in addition to XML schemas, there are only two mandatory documentation artifacts required by every MPD: the *mpd-catalog* and the *change log*. An mpd-catalog (`mpd-catalog.xml`) contains identifiers, key metadata, information, and relationships about the MPD. The change log provides a history of modifications.

A *master document* is also mandatory for IEPDs and EIEMs. These MPD classes are built by different developers, and may be registered into a repository for reuse by many other users and developers; therefore, a minimal form of documentation is absolutely necessary. An IEPD or EIEM master document is the primary source and starting point for human readable documentation (similar to a `readme` file), and should reference (and describe) any other separate documentation artifacts. This requirement ensures that baseline documentation is consistently rooted in a clearly visible artifact within each IEPD and EIEM.

The following subsections will address these artifacts as well as the concepts, metadata, and content each supports.

4.1. Catalog

Every NIEM MPD describes itself through a mandatory mpd-catalog artifact. An mpd-catalog is a multi-purpose XML file containing metadata that describes an MPD's

- Unique identification
- Basic descriptive characteristics
- Directory structure and artifacts
- Lineage and relationships to other MPDs

This metadata is designed to be the minimal required that will facilitate human understanding, tool support, and machine processing. The MPD uses and functions that the mpd-catalog is designed to support include (but are not limited to):

- Automatic identification and processing of artifacts
- Browsing and understanding of MPD artifacts and their content
- Conformance and instance validation (as needed)
- Automatic registration in a registry/repository
- Search, discovery, retrieval of MPDs (for example, through metadata values)
- Reuse of MPDs and their artifacts
- Reuse of Business Information Exchange Components (BIEC) and associated EIEMs
- Tracing and analysis of MPD pedigree

[Rule 4-1]

41 An MPD MUST contain an XML mpd-catalog artifact that validates with the NIEM MPD catalog schema (XSD) and that resides in the root directory of the MPD and bears the file name `mpd-catalog.xml`.

The catalog identifies every artifact, its relative path name, file type, and purpose. This enables a machine to find every artifact regardless of its location within an MPD, and know exactly what it is used for, and therefore, how to process it. [Appendix A: MPD Catalog Schema] defines the structure and semantics of a NIEM `mpd-catalog.xml` file.

4.2. Metadata Concepts

The MPD catalog contains both required and optional metadata for the MPD and its artifacts. The following subsections specify the syntax, formats, and semantics for that metadata.

4.2.1. Version Numbering Scheme

Many published MPDs will be periodically revised and updated; therefore, versioning is required to clearly indicate that changes have occurred. A version number is actually part of the unique identification for an MPD (to be discussed in a subsequent section). For this reason:

[Rule 4-2]

42 Every MPD MUST be assigned a version number.

In order to maintain some consistency while allowing reasonable flexibility to authors, this specification establishes a simple version numbering scheme that is consistent with most common practices. This is the same version numbering scheme that is used for all NIEM releases.

[Rule 4-3]

43 All NIEM version numbers adhere to the regular expression:

```
        version ::= digit+ ('.' digit+)* (status digit+)?
        Where:
                digit   ::= [0-9]
                status  ::= 'alpha' | 'beta' | 'rc' | 'rev'
                'alpha' indicates early development
                'beta' indicates late development; but changing or
                        incomplete
                'rc' indicates release candidate; complete but not
                        approved as operational
                'rev' indicates very minor revision that does not impact
                        schema validation
```

(The regular expression notation used above is from **[W3-XML]** `#sec-notation`)

The regular expression in **[Rule 4-3]** allows the following example version numbers:

- `1`
- `1.2`
- `1.3.1.0`
- `1.2alpha13`
- `199.88.15rev6`

There are two implications in **[Rule 4-3]**. The first is that in some cases this version scheme implies and confirms a chronology of releases. For example, a given product labeled version 2.3 must have been released before the same product labeled 2.3.1. Therefore, version 2.3.1 is more current than version 2.3.

However, this is a multi-series version scheme, and chronological relationships exist only within a given series. So, for example, nothing can be said about a chronological relationship between versions 2.2.4 and 2.3. This is because version 2.2.4 is in a different series (i.e., 2.2) and could actually have been released after 2.3. [Figure 4-1] illustrates a system of versions that uses the numbering scheme of **[Rule 4-3]**.

Figure 4-1. Example versioning system.

[Figure 4-1] illustrates eight different version series. Within this illustration these are the only sequences that have chronological relationships that can be identified through version numbers.

- Series 2: 2.2, 2.3, 2.4

- Series 3: 3.0, 3.1, 3.2
- Series 2.2: 2.2(.0), 2.2.1, 2.2.2, 2.2.3, 2.2.4
- Series 2.3: 2.3(.0), 2.3.1
- Series 2.4: 2.4(.0), 2.4.1
- Series 3.0: 3.0(.0), 3.0.1, 3.0.2
- Series 3.1: 3.1(.0), 3.1.1
- Series 3.2: 3.2(.0), 3.2.1, 3.2.2

The second implication of **[Rule 4-3]** is that pre-releases are easily identified by the strings `alpha`, `beta`, and `rc`. These strings are simple visible indicators of the status or stage of development of an MPD.

This specification places no further restrictions or meaning (implied or otherwise) on a version or release number. Authors have the option to use numbers between dots to indicate degree of compatibility or other relationships between versions as needed. For example, for a given MPD, the author may declare that if an instance validates to version 4.2.3, then it will also validate to version 4.2. Such a claim is acceptable. However, this specification does not imply any such relationships. Any meaning assigned to version sequence by an authoritative source should be unambiguously documented within the MPD.

[Rule 4-4]

44 A higher MPD version number within a version series does NOT imply compatibility between versions. Compatibility between or among MPD versions MUST be explicitly stated in documentation.

Note that an author who updates an existing MPD to a new version for no other reason than to conform to this specification may choose the version number based on its previous version number or not, as long as it follows the version number syntax.

4.2.2. URI Scheme for MPDs

To facilitate MPD sharing and reuse the assignment of a URI (Uniform Resource Identifier) to each MPD is essential.

[Rule 4-5]

451 Every MPD MUST be assigned a valid `http` URI.

This specification follows **[RFC3986 URI]**, which defines the syntax and format for a URI. However, this specification also restricts an MPD URI to a URL and does not allow a URN (Uniform Resource Name) to be assigned to an MPD.

Here is a typical example of an http URI: `http://www.abc.org/niem-iepd/order/2.1.2rev3/`

Note that **[Rule 4-5]** explicitly states that a URI assigned to an MPD must be valid. This means that the person or organization assigning the URI either is the registrant of the domain name, or has authority from the registrant to assign this URL as an MPD URI. In the example above, `www.abc.org` is the domain name (between the second and third "/"). There is no requirement for a URL assigned to an MPD to resolve to any particular Internet resource or to resolve at all. However, it is always good practice for such a URL to resolve to the resource it represents, the directory it resides in, or to documentation for that resource. See `http://www.w3.org/Provider/Style/URI.html`

The MPD version number is essential to its unique identification. Incorporation of the version number within the MPD URI provides a simple visual (as well as machine readable) means of identifying one of the most fundamental relationships between MPDs, i.e., that one is a different version of another.

[Rule 4-6]

452 The URI for an MPD MUST end in its version number.

Another advantage to this technique is that different versions of an MPD will generally group together in a standard sorted ordering. (Of course, this assumes that a related family of MPDs follows the same URI scheme.)

And finally, note that `mpd-catalog.xsd` defines a mandatory attribute for both the `mpdURI` and the `mpdVersionID`. Since the ending string of an MPD URI must be its version ID, then this means that the catalog duplicates the MPD version ID in two locations. This is by design. You will discover in Section 6.1, *MPD File Name Syntax*, below, that MPD file name syntax also intentionally duplicates the catalog `mpdName` and `mpdVersionID`. There are two reasons for this design. First, software tools are expected to build and process MPD catalogs. Instead of forcing tool developers to parse the MPD URI just to retrieve the version ID, the catalog provides a separate `mpdVersionID` attribute. Second, duplication of key metadata in URIs and file names facilitates faster visual recognition of an MPD, rather than requiring that a user open an its archive, open its `mpd-catalog.xml`, and scan its content just to locate `mpdName` or `mpdVersionID`.

4.2.3. URI Scheme for MPD Artifacts

Given the URI for an MPD, a URI exists for each artifact in that MPD. Again, this specification follows **[RFC3986 URI]** and employs a fragment identifier to designate an artifact URI.

The `mpd-catalog.xsd` schema in [Appendix A: MPD Catalog Schema] declares an `id` attribute of type `xs:ID` that is required for use in `FileType` and `FileSetType`; optional in `FolderType`. Within `mpd-catalog.xml` an MPD author must assign a locally unique `id` value to each artifact and artifact set of the MPD. A globally unique URI for an artifact is the concatenation of the MPD URI with "#" followed by the `id` value of the artifact or artifact set.

Since every MPD must have a URI, and an MPD catalog must list all artifacts contained in the MPD, and each artifact must be assigned a locally unique id value, then each MPD artifact has a globally unique URI that can be referenced from other external resources as needed. The following rules concerning an artifact `id` apply:

[Rule 4-7]

46 Each file artifact in an MPD MUST have a corresponding File element in its mpd-catalog.

[Rule 4-8]

47 Each file set artifact in an MPD MUST have a corresponding FileSet element in its mpd-catalog. This FileSet element must identify each file artifact that is a member of that file set artifact.

These rules and the catalog schema specified in Appendix A: MPD Catalog Schema require that each individual file artifact and each set of file artifacts grouped for a particular purpose must be identified in the catalog. This is to facilitate automatic processing of the MPD by software tools. Note that file subdirectories (or folders) are independent of file set grouping. Each file artifact can be identified in one and only one subdirectory (or folder) by its relative path name and file name. However, each file artifact can be a member of multiple file set artifacts. Therefore, while it is possible to associate a file set artifact with a single subdirectory within an MPD, it is not required.

Also note that the `Folder` element in [Appendix A: MPD Catalog Schema] is available to represent subdirectories in MPDs. This allows an XSLT to generate a catalog index that can display subdirectories as needed. However, operating system subdirectories are not authoritative for file artifact and file set organization. For this reason the Folder element contains an optional `id` attribute, a required `relativePathName` attribute, but no URI attributes that could enable file grouping.

[Rule 4-9]

48 Each artifact identified in the mpd-catalog MUST be assigned an id in the format of an NCName (Non-Colonized Name) as defined by **[W3-XML-Namespaces]**. This is required for both File and FileSet artifacts.

By the rules of **[W3-XML]** the value of each id attribute (which is of type `xs:ID`) must be locally unique within the `mpd-catalog.xml` file. However, globally unique URIs are required to identify and reference artifacts between MPDs. To facilitate references from one MPD catalog to another, the following rule applies:

[Rule 4-10]

49 A URI reference to an individual MPD artifact from another resource is the concatenation of

- The URI of the MPD that contains the artifact.
- The crosshatch or pound character ("#").
- A fragment identifier that is the locally unique `id` of the artifact within the mpd-catalog of the MPD itself.

Example of an artifact URI: `http://www.abc.org/niem-iepd/order/2.1.2rev3/#a552`

To illustrate a typical scenario for using this URI, a developer can build an IEPD that contains a schema artifact within the catalog to which he assigns: `id="x25"` (a locally unique `id` within the IEPD catalog). If this schema artifact is a duplicate of the `a552` artifact from the published MPD whose URI is `http://www.abc.org/niem-iepd/order/2.1.2rev3/`, then the developer can optionally assign the following attribute to this artifact's catalog entry:

```
externalURI="http://www.abc.org/niem-iepd/order/2.1.2rev3/#a552"
```

Additional `externalURI` attributes can be assigned to this artifact if the author knows of other reuses of this same artifact in other MPDs.

4.2.3.1. MPD Artifact URIs Are Not NIEM Namespaces

URI does not have the same meaning as namespace. Do not rely on namespaces for artifact URIs. Recall that the namespaces used in a schema subset derived from a NIEM release are identical to the namespaces of the release itself. Furthermore, an IEPD or an EIEM may contain multiple subsets. The non-uniqueness of NIEM namespaces implies that they cannot be used as URIs for MPD artifacts.

[Rule 4-11]

410 NIEM namespaces MUST NOT be used as URIs for MPD artifacts.

4.2.4. Wantlists

A NIEM subset is often associated with a NIEM *wantlist*. A *wantlist* is an abbreviated representation of a NIEM schema subset, and identifies only the data components a user selected (as requirements) to build that subset. However, there are usually a number of additional data components that the user selections are dependent upon and therefore must be added to construct the complete schema subset. For example, a user may decide that he/she requires and therefore select `nc:Person`. In this case, the wantlist will only contain that component, but the associated subset must contain both `nc:Person` and `nc:PersonType`. A software tool that understands how to process NIEM subsets and wantlists (such as the NIEM Schema Subset Generator Tool **[NIEM SSGT]**) will ensure correct correlation between a wantlist and a subset.

[Definition: wantlist]

An XML document that represents a NIEM schema subset. A NIEM wantlist identifies the data component requirements declared by the author of a subset; it does not identify the data component dependencies required to reconstitute the complete subset. The complete subset can be computed with the reference schema from which the subset was derived.

A wantlist is always associated with a subset schema set. Furthermore, a wantlist may also be associated with a constraint schema set, because this is often built by starting with a subset. For a simple IEPD, it is often trivial to identify a single subset with its corresponding wantlist within the `mpd-catalog.xml`. On the other hand, the MPD Specification does not prevent developers from building complex IEPDs that contain: (1) a base schema set supported by multiple subsets and associated wantlists, and (2) multiple constraint schema sets, each supported by multiple wantlists. As with other complex cases, the developer is responsible to clearly document the associations between wantlists and the schemas they support. In order to maintain a minimal degree of consistency for placement of a wantlist within a catalog (especially in the case of simple IEPDs):

[Rule 4-12]

411 A wantlist in an `mpd-catalog.xml` MUST be a member of the base or constraint schema set it is associated with.

4.2.5. MPD Artifact Lineage

An important MPD business requirement is transparency of lineage. It must be possible to easily identify the relationships that may exist among families, versions, adaptations, specializations, generalizations, etc. of MPDs. The established URI scheme for MPDs and MPD artifacts as well as the catalog help make this possible.

The catalog provides a `Relationship` element with three attributes (`resourceURI`, `relationshipCode`, and `descriptionText`) to identify the pedigree of an MPD. There are many ways that one MPD may relate to another. This makes it extremely difficult to specify a fixed set of values that could objectively define an exact relationship between a pair of MPDs. Therefore, the optional `descriptionText` attribute is provided to further explain the nature of any of the eight `relationshipCode` values available (`version_of`, `specializes`, `generalizes`, `deprecates`, `supersedes`, `adapts`, `conforms_to`, `updates`). In some cases, the value of `relationshipCode` may be generic enough to require a more detailed explanation in `descriptionText` (for example, if the value is `"adapts"`).

The catalog also enables an MPD author to record a fine-grained pedigree between MPDs when reusing artifacts from other MPDs. By default each artifact identified in a catalog has a globally unique URI (using a fragment reference) that can refer to it. An MPD author signifies reuse of a given artifact by entering the URI for that artifact in the optional `externalURI` attribute within the `File` element. Use of the `externalURI` for a given artifact does not preclude the mandatory requirement to assign a locally unique `id` to that artifact (per **[Rule 4-8]** and **[Rule 4-9]**).

Note that some MPDs are designed for more extensive reuse than others. For example, many IEPDs are expected to reuse an EIEM. In such cases, the catalogs for these IEPDs and the corresponding

EIEM may overlap in or duplicate a large number of metadata and references. This is expected. The catalog contains many references to and semantics for artifacts and MPDs. Correct and consistent use of these references and semantics will create networks of related MPDs so that tools can automatically locate, parse, and process MPDs and their corresponding artifacts as needed and when available in shared repositories.

4.3. Change Log

4.3.1. NIEM Releases, Core Updates, and Domain Updates

Although the version identifier is useful for a fast and visual indication of the state of an MPD relative to others, it only provides a fairly generalized indication of the volume, complexity, and impact of changes that have been applied since a previous version. Of course, a change log is required to ensure a more specific accounting of changes from one version to another.

Once published, NIEM releases always exist. This ensures that IEPDs and EIEMs built from them will always be stable, and may be updated to a new NIEM release only when convenient or absolutely necessary to take advantage of new or modified data components. Though not recommended, the NIEM program does not prohibit a developer from building an IEPD based on a NIEM release that is older than the most current version. There may be potential disadvantages related to interoperability levels achievable with others developing to the latest release. Nonetheless, an older version might meet the business needs of a particular organization quite well.

In spite of this built-in stability, the NIEM architecture is designed to evolve as requirements change. New versions of reference schema sets such as NIEM releases, core updates, and domain updates can have significant impacts on future IEPDs and EIEMs. Developers must understand in detail how changes will affect their IEPD and EIEM products and the tools used to build them. To work effectively, tools for domain content development, impact analysis, migration between releases, etc. must be able to digest formal change logs. A formal change log is also essential to efficiently process and integrate new and changed content into NIEM for new releases, and to simultaneously maintain multiple versions of NIEM for users. All of the foregoing reasons dictate that NIEM require a normative change log for reference schema sets.

[Rule 4-13]

412 Every MPD that is a reference schema set (i.e., NIEM releases, core updates, and domain updates) MUST contain an XML change log artifact that:

- Validates with the NIEM change log schemas `mpd-changelog.xsd` and `niem-model.xsd`) Note these are the base filenames; the actual filenames also contain a version number; for example, `mpd-changelog-1.0.xsd`.
- Records changes to previous reference schemas that this MPD represents.
- Bears the file name `changelog.xml`.
- Resides in the root directory of the MPD.

The current version of `mpd-changelog.xsd` can be found at:

`http://reference.niem.gov/niem/resource/mpd/changelog/`

The current version of `niem-model.xsd` which describes the NIEM conceptual model can be found at:

`http://reference.niem.gov/niem/resource/model/`

Since the schemas are the authority for a release or update and because almost all tool support depends on the schemas, the change log is only designed to audit transactional change to the schemas in the reference set. There is no provision for logging changes to support documentation or other non-schema artifacts. Non-schema changes are handled non-normatively in the form of release notes.

4.3.2. IEPDs and EIEMs

IEPD and EIEM change log requirements are less strict and are not required to conform to the naming and schema specifications in **[Rule 4-13]**. However, a change log is still required.

[Rule 4-14]

413 Every MPD that is an IEPD or EIEM MUST contain a change log artifact that:

- Records changes to previous IEPD or EIEM schemas that this MPD represents.
- Begins with the substring "changelog".
- Resides in the root directory of the MPD.

This rule does not specify the format for an IEPD or EIEM change log. This is left to the discretion of the author. While use of `mpd-changelog.xsd` is encouraged for IEPD and EIEM schemas, it is not required. Relaxing the change log format encourages and facilitates easier and more rapid development. IEPDs and EIEMs are developed by a variety of NIEM domains, organizations, and users; and they are intended to specify implementable exchanges. As a result, IEPDs and EIEMs usually contain many documentation artifacts as well as various machine readable artifacts in various formats. A consistent standard change log for these artifacts is very difficult to specify strictly.

The initial version of an IEPD or EIEM would not likely require a change log. However, for consistency of validation and to help facilitate automatic processing of IEPDs and EIEMs by tools:

[Rule 4-15]

414 The initial version of an IEPD or EIEM MUST contain a change log artifact with at least one entry for its creation date.

Finally, if the `mpd-changelog.xsd` specification is used for IEPD/EIEM schema changes, then it is potentially possible that such an MPD will need a second change log if the author wants to accommodate documentation or other changes not related to schemas (since `mpd-changelog.xsd` cannot be extended to accommodate such changes). If this is the case, then the following rule applies:

[Rule 4-16]

415 If an IEPD or EIEM contains more than one change log artifact, then each change log artifact MUST:

- Have a file name that begins with the substring `changelog`.
- Reside in the MPD root directory.

4.4. Master Document

A master document is only required for IEPDs and EIEMs since these MPDs are allowed the greatest design flexibility, can be developed and implemented different ways, and are not centrally managed. On the other hand, releases and domain updates have fairly restrictive rules to obey, standard documentation for how to use them, and are centrally managed.

[Rule 4-17]

416 An IEPD or an EIEM MUST contain a master document located in the MPD root directory whose filename begins with the substring `master-document`.

The master document may replicate some of the metadata in the mpd-catalog. However, the mpd-catalog is intentionally designed to be efficient, easily to parse, and minimal. It is intended for search, discovery, registration, and Web page generation, and not to support various types of detailed technical prose often required for human understanding.

The primary purposes of the master document include:

- To help facilitate understanding and reuse of IEPDs and EIEMs.
- To ensure that fundamental and detailed business-level information about an IEPD or EIEM are documented for human understanding.
- To ensure an author has considered and conveys such fundamental information.
- To provide an initial source within an IEPD or EIEM for human consumable documentation (similar to a "readme" file) and/or references to other business or technical documentation needed for understanding.

The master document is not intended to be the only source of written documentation for an MPD (though it can be). It is expected to be the initial resource that references and coordinates all others whether physically present in the MPD or linked by reference. Consider the master document to be similar to a `readme` file. Many organizations have their own customized formats and operating procedures for documenting their work and products. This specification does not attempt to standardize master document format or layout. Only the file name and relative path within the MPD are strictly specified. The following section will also describe in general terms minimal content that should be in the master document. Adherence to such a requirement is certainly a subjective judgment.

4.4.1. Master Document Content

This section is neither a cookbook nor a normative specification for a master document. It simply suggests typical topics that a master document should or might address, and provides some non-normative guidance.

The master document should help another user or developer to understand the content and use of an IEPD or EIEM, as well as determine potential for reuse or adaptation. It should describe what implementers need to understand and what the author considers is important to understanding an IEPD or EIEM. There is no limit or constraint on its content.

At a minimum, the master document should contain several fundamental elements of information about the MPD:

- Purpose of this MPD.
- Scope of its deployment, usage, and information content.
- Business value and rationale for developing it.
- Type of information it is intended to exchange (in business terms).
- Identification of senders and receivers (or the types of senders and receivers).
- Typical interactions between senders, receivers, and systems.
- References to other documentation within the MPD, and links to external documents that may be needed to understand and implement it.

[Rule 4-18]

417 A NIEM IEPD or EIEM master document SHOULD (at a minimum) describe the MPD purpose, scope, business value, exchange information, senders/receivers, interactions, and references to other documentation.

MPD documentation types and formats will vary with the methodologies and tools used to develop them. Most of this documentation will likely be typical of that generated for data-oriented software projects. Some documentation may only require sections in the master document. Other documentation may be more suitable as separate artifacts that are referenced and explained by a section in the master document (such as diagrams, large tables, data dictionaries, test results/reports, etc.). The following are some common examples of sections in or separate artifacts associated with the master document:

- Use cases
- Business processes
- Business requirements>
- Business rules
- Metadata security considerations
- Domain model design specifications and documentation and/or diagrams
- Data dictionary
- Testing and conformance
- Development tools and methodologies used
- Implementation guidance (An IEPD is meant to be implemented, so this section is very important, particularly in the case of a complex IEPD that uses multiple subsets, exchange schemas, and/or IEP root elements.)
- Security considerations (for protecting sensitive or classified information)
- Privacy considerations (for example, Personal Identifiable Information or PPI)
- Types of implementation
- If an IEPD employs multiple subsets (either from the same release or from different releases):
- Where and how are these used?
- What are the caveats regarding duplicative data components?
- How are these coordinated in the implementation?
- If an IEPD employs multiple exchange schemas and/or exchange schemas with multiple root elements:
- What is the purpose of each (exchange and root) and when is it used?
- How are these coordinated during the runtime preparation and transmission of IEPs?
- Authors are also encouraged to include an executive summary, especially for particularly lengthy master documents.

5. Optional MPD Artifacts

Aside from the required artifacts, MPD content is relatively flexible. A variety of other optional documentation files may be incorporated into an MPD. When applicable, these may include (but are not limited to) files that describe or explain:

- Implementation details (hardware, software, configuration, etc.)

- Use of multiple root elements
- Use of multiple subsets or mixed releases
- How to use/reuse an MPD for various purposes (such as Web Services)
- Rationales and/or business purposes

In addition to documentation artifacts, a variety of other optional files can be added to an MPD to facilitate tool support and make reuse, adaptation, and/or implementation easier. These are often files that are inputs to or outputs from software tools. Examples include content diagrams, content models in tool-specific formats, and business rules (either formal or informal representations).

Another optional artifact that is encouraged, especially for IEPDs, is a conformance report or other evidence of quality. In the future, as NIEM processes and tools mature, a conformance and quality reports and a corresponding certificate may become required artifacts. For now, inclusion of a conformance report is at the discretion of the author or sponsor. Though clearly, such reports can only increase confidence in MPDs that contain them.

An MPD author may include any files believed to be useful to understand, implement, reuse, and/or adapt an MPD. However, [Rule 4-6] always applies any file included as an artifact in an MPD must also be identified with an entry in the mpd-catalog.

An MPD of relatively simple content and scope may only need to contain the minimum mandatory artifacts required by this specification in order to understand and implement it. (See [Appendix F: MPD Artifacts] for a listing of the mandatory and optional artifacts for each type of MPD.)

Files vary widely in format and are often specific to the tools an author uses to parse, consume, or output them. Therefore, if tool-specific files are included in an MPD, it is also a good practice to include copies of those files in formats that display with standard Web browsers or other cost-free, publicly available viewing tools. Common formats include, but are not limited to ASCII text, CSV, HTML, XHTML, JPG, GIF, PNG, and PDF. This guidance is intended to encourage and facilitate maximal sharing and distribution of MPDs; it does not prohibit and is not intended to discourage the inclusion of other file formats.

In particular, this specification does not discourage use of Microsoft (MS) file formats for documentation and other optional artifacts. A number of MS Office products are commonly used in most large organizations. Furthermore, viewers are publicly available for many MS products, including Word (DOC), Excel (XLS), PowerPoint (PPT), Access (MDB), and Visio (VSD). (See `http://office.microsoft.com/en-us/downloads/office-online-file-converters-and-viewers-HA001044981.aspx`

6. Directory Organization, Packaging, Other Criteria

An MPD is a logical set of electronic files aggregated and organized to fulfill a specific purpose in NIEM. Directory organization and packaging of an MPD should be designed around major themes in NIEM: reuse, sharing, interoperability, and efficiency.

This rule is also applicable to all MPDs:

[Rule 6-1]

61 An MPD is packaged as a single compressed archive of files that represents a sub-tree of a file system in standard **[PKZIP]** format. This archive MUST preserve and store the logical directory structure intended by its author.

MPD NIEM schema artifacts must be valid for both XML Schema and NIEM:

[Rule 6-2]

62 Within an MPD archive, all XSD and XML artifacts MUST be valid against and follow all rules for their respective **[NIEM NDR]** conformance targets (i.e., subset, constraint, extension, exchange, reference schema documents, as well as instance XML documents); this includes being well-formed and valid XML Schema documents.

NIEM releases, core updates, and domain updates follow a relatively consistent approach to directory organization **[NIEM Domain Update Specification]**. But there are many ways to organize IEPD and EIEM directories that may depend on a number of factors including (not limited to) business purpose and complexity. For this reason, strict rules for IEPD and EIEM directory structure are difficult to establish. Therefore, IEPD and EIEM authors may create their own logical directory structures. However, for consistency and efficiency:

[Definition: MPD root directory]

The top level file directory relative to all MPD artifacts and subdirectories.

[Rule 6-3]

63 An MPD archive MUST uncompress (unzip) to a one and only one MPD root directory.

[Rule 6-3] ensures that:

- Unpacking an MPD archive will not scatter its contents on a storage device.
- A common starting point always exists to explore or use any MPD.
- mpd-catalog and change log artifacts will always be found in the MPD root directory (as a result of [Rule 4-1] and [Rule 4-12]).

6.1. MPD File Name Syntax

As previously stated, the MPD Specification is intended to facilitate tool support for processing MPDs. Given a tool must process an MPD, providing it basic information about the MPD as early as possible will help to reduce processing time and complexity. So, if the class and version of an MPD archive could be easily identified by its file name, then a tool would not have to immediately open the archive and process the mpd-catalog just to determine this information. Of course, ultimately, to do anything useful, a tool will have to open the MPD archive and process its mpd-catalog. However, a standard file name syntax would allow a tool to search through a set of MPD archives to find a particular MPD name, version, or class without having to open (unzip) each. The following rules apply:

[Rule 6-4]

64 An MPD archive file MUST use file name syntax defined by the regular expression:

```
mpd-filename ::= name '-' version '.' class '.zip'
Where:
        name     ::= alphanum ((alphanum | special)* alphanum)?
        alphanum ::= [a-zA-Z0-9]
        special  ::= '.' | '-' | '_'
        version  ::= digit+ ('.' digit+)* (status digit+)?
        digit    ::= [0-9]
        status   ::= 'alpha' | 'beta' | 'rc' | 'rev'
        class    ::= 'rel' | 'cu' | 'du' | 'iepd' | 'eiem'
```

All alpha characters SHOULD be lower case to reduce the risk of complications across various file systems. See [Rule 4-3] for an explanation of the status options.

(The regular expression notation used above is from **[W3-XML]** `#sec-notation`.)

Obviously, the set of class values corresponds to release, core update, domain update, IEPD, and EIEM respectively. A valid IEPD file name corresponding to the example in [Appendix C: Sample MPD Catalog Instance] would be: `Planning_Order-1.0.3rev2.iepd.zip`

Checking this Appendix you will find that this example obeys the following two rules:

[Rule 6-5]

65 Within an MPD, the `name` and `version` substrings in the file name MUST match exactly the values for attributes `mpdName` and `mpdVersionID` within its `mpd-catalog.xml` artifact.

[Rule 6-6]

66 Within an MPD, the `class` substring in the file name MUST equal the `mpdClassCode` attribute value within the mpd-catalog. Values are:

```
rel     = release
cu      = core update
du      = domain update
iepd    = information exchange package documentation
```

```
        eiem    = enterprise information exchange model
```

In HTTP-based Web Services environments, the MIME type designation of a MPD archive is important to facilitate processing by service consumers.

[Rule 6-7]

67 When represented on the Internet, an MPD archive SHOULD use the following MIME Type:

```
        application/zip+class
                where
                "class" is one member from the set {rel, cu, du, iepd,
eiem}.
```

Use of the generic zip MIME type `application/zip` is allowed, but discouraged. No other MIME types are allowed when representing MPD archives.

6.2. Artifact Links to Other Resources

The **[NIEM NDR]** requires that each `xs:import` in a NIEM schema contain a schemaLocation attribute with either an absolute or relative path reference that resolves to the correct imported schema. However, this specification restricts an MPD import to a relative path reference that resolves to the correct schema within the MPD itself. It is important to understand that the URI scheme previously discussed in [Section 4.2.3 URI Scheme for MPD Artifacts] should be used only to identify relationships among and provide source links to external schemas being reused. It is not sufficient to allow references or links to such schemas stand in for a physical copy.

Regardless of references to external sources or internal MPD directory organization, each schema `xs:import` must adhere to the following rule:

[Rule 6-8]

68 Within an MPD archive, the value of each `xs:import schemaLocation` attribute MUST be a relative path reference that resolves to the correct schema within the sub-tree.

The implication of this rule is a guarantee that all schema artifacts necessary to define, validate, and use an MPD are physically present within that MPD without requiring modifications to `xs:import schemaLocation` attributes within schemas. In accordance with the **[NIEM NDR]**, if MPD schemas are moved to an operational environment for implementation, validation, or other purposes, then absolute references may replace relative path references when needed. When absolute references to Internet resources are required:

[Rule 6-9]

69 Absolute references to Internet resources MUST use a well-known transfer protocol (http, https, ftp, ftps) and MUST resolve (If applicable, documentation that describes how to resolve with security, account, and/or password issues MUST be included).

Releases, core updates, and domain updates must adhere to packaging rules primarily to enable development tools to process them consistently and efficiently. The NIEM PMO controls the format and documentation for these MPDs and publishes them at `http://release.niem.gov/niem/`. However, many different organizations author IEPDs and EIEMs. As such, they may be distributed, published in repositories (possibly to a limited community), and reused by others. Furthermore, EIEMs are the basis for families of IEPDs. Therefore, it is important that both of these MPD classes are well documented for understanding and use.

[Rule 6-10]

610 A published IEPD MUST contain all documents necessary to understand it and allow it to be implemented correctly.

[Rule 6-11]

611 A published IEPD MUST link (through its mpd-catalog) to any EIEM it is based on. When represented on the Internet, an MPD archive SHOULD use the following MIME Type:

```
        application/zip+[class]
                where
                [class] is one member from the set {rel, cu, du, iepd,
eiem}
```

Use of the generic zip MIME type `application/zip` is allowed, but discouraged. No other MIME types are allowed when representing MPD archives.

The **[NIEM NDR]** explains how NIEM employs a special type adaption mechanism to encapsulate and use other standards (e.g., geospatial and emergency management standards) in their native forms that are not NIEM-conformant. Other standards may use `xs:import` without requiring `schemaLocation` attributes (instead, relying only on the namespace). These standards may also use `xs:include` which is disallowed by NIEM. When standards external to NIEM are required within NIEM MPDs, the following rule applies:

[Rule 6-12]

612 Within an MPD archive, if non-NIEM-conformant schemas from other standards are used and referenced within an MPD, then all `xs:import`, `xs:include`, and `xs:redefine` constructs used within those schemas MUST be modified as needed to have a value for the schemaLocation attribute that is a relative path reference that resolves to the correct schema within the sub-tree.

For the case of non-NIEM-conformant schemas, this rule ensures that all schemas (or corresponding artifacts and namespaces) from other standards required for definition, validation, and use of the MPD are present within the archive.

XML schemas are the heart of MPDs since they formally specify normative structure and semantics for data components. However, in general, an MPD is a closed set of artifacts. This means that all hyperlink references within artifacts should resolve to the appropriate artifact.

[Rule 6-13]

613 Within any artifact of an MPD archive, any direct reference to another resource (i.e., another artifact such as an image, schema, stylesheet, etc.) that is required to process or display an artifact SHOULD exist within the archive at the location specified by that reference.

This means that MPD artifacts, including documentation artifacts, should be complete. For example, if an HTML document contains a hyperlink reference (`href`) to a schema (xsd) or stylesheet (xsl) that is part of the MPD, then the schema file associated with that hyperlink should be present within the MPD; likewise for a sourced (`src`) image. Authors should exercise good judgment with this rule. For example, it does not require an MPD to contain copies of all cited documents from a table of references if it contains hyperlinks to those documents. The key operating words in this rule are: "another resource is required to process or display an artifact SHOULD exist within the archive."

6.3. Duplication of Artifacts

Within an MPD, the replication of files or entire file sets should be avoided. However, replication is allowed when a reasonable rationale exists. In some cases, file replication may make it easier to use, validate, implement, or automatically process an MPD. For example, multiple subsets may overlap with many identical schemas. Yet, it may be easier or even necessary to allow this form of duplication to accommodate a validation tool, rather than removing duplicate schemas, and forcing the tool to use the mpd-catalog to identify required artifacts. `mpd-catalog-1.0.xsd` is designed to track duplicate artifacts (File or FileSet), as well as, reference a single File artifact from multiple `FileSet` artifacts.

6.4. Non-normative Guidance for Directories

Aside from the rules above, this specification does not impose additional constraints on an IEPD or EIEM author's freedom to organize directory structure. This is why the mpd-catalog is required to list every artifact, along with its locally unique identifier, relative path name, nature, and purpose. This enables a machine to find every artifact regardless of its location in an MPD archive and know exactly what it is. The mpd-catalog artifact always takes precedence over the directory structure of an IEPD or EIEM.

Non-normative guidance for directory structuring may be useful to authors for a relatively simple IEPD or EIEM with a single subset, extension set, constraint set, and exchange schema set. The following general guidance has been common practice for IEPD directories:

- Create a root directory for the IEPD from the name and version identifier of the IEPD. For example `my-iepd-3.2.4rev2`.

- Per [Rule 4-1] and [Rule 4-12], the mpd-catalog and the change log must reside in the root directory.
- Maintain XML stylesheets used with the mpd-catalog and change log in the MPD root directory.
- Maintain each subset organized as generated by the Schema Subset Generation Tool (SSGT). The reason for maintaining the SSGT grouping is that the SSGT ensures that all `xs:import schemaLocation` attributes contain relative path names that are correctly coordinated with the directory structure.
- If derived from a schema subset, maintain the constraint schema set grouped as the subset from which it was derived (for the same reason as above).
- Establish a subdirectory of the MPD root directory with the name `XMLschemas`. Within this subdirectory:
    - Maintain each subset in a subdirectory with a name that contains the substring `subset`.
    - Maintain and correlate a wantlist with the subset it represents. To do so, change a wantlist filename if appropriate.
    - Maintain each constraint schema set (or all constraint schema documents if appropriate) in a subdirectory with a name that contains the substring `constraint`.
    - Maintain extension schema documents in a subdirectory with a name that contains the substring `extension`.
    - Maintain exchange schema documents in a subdirectory with a name that contains the substring `exchange`.
- Maintain all documentation in a subdirectory of the MPD root directory with a name that contains the substring `documentation`. Create additional documentation subdirectories inside this one as needed.
- Maintain all sample XML instances that validate to the schemas in a subdirectory of the MPD root directory with a name that contains the substring `XMLsamples`. This subdirectory can also contain any XML stylesheets (XSL) used with the sample instances.
- Maintain tool specific files (inputs and outputs) in a subdirectory of the MPD root directory with a name that contains the substring `library`.

The guidance above results in an IEPD directory structure illustrated below in Figure 6-1. Obviously, there are many other ways to organize for more complex business requirements in which multiple releases, subsets, constraint sets, core updates, and domain updates are employed in a single IEPD.

Figure 6-1: 6-1. IEPD sample directory structure.

Appendix A. Acronyms and Abbreviations

API	Application Programming Interface
BIEC	Business Information Exchange Component
CSV	Comma Separated Value (file format)
EIEM	Enterprise Information Exchange Model
GIF	Graphic Interchange Format
HLTA	High-Level Tool Architecture
HLVA	High-Level Version Architecture
HTML	Hyper Text Markup Language
IEP	Information Exchange Package
IEPD	Information Exchange Package Documentation
JPG/JPEG	Joint Photographic (Experts) Group
MPD	Model Package Description
NA	Not Applicable
NDR	Naming and Design Rules
NIEM	National Information Exchange Model
NTAC	NIEM Technical Architecture Committee
PDF	Portable Document Format
PMO	Program Management Office
RDF	Resource Description Framework
SSGT	Schema Subset Generation Tool
SVG	Scalable Vector Graphics
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XHTML	Extensible Hyper Text Markup Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

Appendix B. Glossary of Terms

The following terms are defined in the context of NIEM.

artifact A single file with a defined purpose or a set of files logically grouped for a defined purpose. An MPD is a collection of artifacts, the purpose for which is to define and document the intended use of the MPD.

base schema document set A NIEM MPD artifact that is the set of all NIEM-conformant and external non-conformant XML schemas that together specify an information exchange or information model for an MPD. A base schema document set may incorporate reference, subset, extension, and exchange schema documents, as well as external schema documents from authoritative sources outside of NIEM.

Business Information Exchange Component (BIEC) A NIEM-conformant XML schema data component definition or declaration (for a type, element, attribute, or other XML construct) reused, subsetted, extended, and/or created from NIEM that meets a particular recurring business requirement for an information sharing enterprise.

constraint schema document A schema document which imposes additional constraints on NIEM-conformant instances. A constraint schema document or a constraint schema document set validates additional constraints imposed on an XML instance only after it is known to be NIEM-conformant (i.e., has been validated to reference schema documents, subset schema documents, extension schema documents, and/or exchange schema documents). Constraint schema validation is a second-pass validation that occurs independently of and after conformance validation. A constraint schema need not validate constraints that are applied by other schemas. See also constraint schema document set.

constraint schema document set A set of related constraint schema documents that work together, such as a constraint schema document set built by adding constraints to a schema document subset. See also constraint schema document.

core update An MPD that applies changes to a given NIEM core schema document or document set. It never replaces a NIEM core; instead, it is used to add new schema documents, new data components, new code values, etc. to a particular NIEM core. In some cases, a core update can make minor modifications to existing core data components.

data component An XML Schema type or attribute group definition; or an XML Schema element or attribute declaration.

domain update A MPD that contains a reference schema document or document set issued by one or more domains that constitutes new content or an update to content that was previously included in a NIEM release. A domain update may define and declare new versions of content for NIEM releases or other published content. The issuing body vets each update before publishing, but the update is not subject to review by other NIEM bodies. A domain update must be NIEM-conformant, but otherwise it has fewer constraints on quality than does a NIEM release. Domain update schema documents contain proposed future changes to NIEM that have not been published in a numbered release and have not been vetted by NIEM governance bodies (except by the domain or domains involved). Domain updates are published to the NIEM Publication Area at http://publication.niem.gov/ and available for immediate use within IEPDs.

Enterprise Information Exchange Model (EIEM) An MPD that contains a NIEM-conformant schema document set that defines and declares data components to be consistently reused in the IEPDs of an enterprise. An EIEM is a collection of BIECs organized into a schema document subset and one or more extension schema documents. Constraint schema documents and non-NIEM-conformant external standards schema documents with type adapters are optional in an EIEM.

exchange schema document A NIEM-conformant schema document that declares an XML document element (top-level) for a particular information exchange.

extension schema document A NIEM-conformant schema document that adds domain or application specific content to the base NIEM model.

harmonization Given a data model, harmonization is the process of reviewing its existing data definitions and declarations; reviewing how it structures and represents data; integrating new data components; and refactoring data components as necessary to remove (or reduce to the maximum extent) semantic duplication and/or semantic overlap among all data structures and definitions resulting in representational quality improvements.

IEP root element The single top-level element in an IEP (instance XML document). In the absence of any other XML wrapping of an IEP, a root element declared in an exchange schema document is an IEP document element.

Information Exchange Package (IEP) An XML document that is a valid instantiation of a NIEM IEPD, and therefore, validates with the schema document set of that IEPD.

Information Exchange Package Documentation (IEPD) An MPD that defines one or more (generally recurring) XML data exchanges.

Information Sharing Enterprise A group of organizations with business interactions that agree to exchange information, often using multiple types of information exchanges. The member organizations have similar business definitions for objects used in an information exchange and can usually agree on their common BIEC names and definitions.

major release A NIEM release in which the NIEM Core reference schema document has changed since previous releases. The first integer of the version number indicates the major release series; for example, versions 1.0, 2.0, and 3.0 are different major releases.

micro release A NIEM release in which neither the NIEM Core nor the domain reference schema documents have changed from the previous major or minor release, but one or more new reference schema documents have been added (without impact to domain or Core schemas). A third digit greater than zero in the version number indicates a micro release (for example, v2.1.1 note that this release does not exist as of this date).

minor release A NIEM release in which the NIEM Core has not changed from previous releases in the series, but at least one or more domain reference schema documents have changed. A second digit greater than zero in the version number indicates a minor release (for example, v2.1). Note also that major v2.0 and minor v2.1 are in the same series (i.e., series 2) and contain the same NIEM Core schema document.

Model Package Description (MPD) A set of related W3C XML Schema documents and other supporting files organized as one of the five classes of NIEM schema sets:

Release (major, minor, or micro).
Domain update (to a release).
Core update (to a release).
Information Exchange Package Documentation (IEPD).
Enterprise Information Exchange Model (EIEM).

An MPD is self-documenting and provides sufficient normative and non-normative information to allow technical personnel to understand how to use or implement it. An MPD is packaged as a ZIP **[PKZIP]** file.

MPD root directory The top level file directory relative to all MPD artifacts and subdirectories.

nature An indication of the type of an MPD artifact. This further indicates how it should be processed by software tools.

purpose A property of an MPD artifact that indicates its usage or function. This determines what to do with this artifact and/or how it should be processed by software tools.

reference schema document An XML Schema document that meets all of the following criteria:

It is a conformant schema document.

It is explicitly designated as a reference schema document. This may be declared by an MPD catalog or by a tool-specific mechanism outside the schema document.

It provides the broadest, most fundamental definitions of components in its namespace.

It provides the authoritative definition of business semantics for components in its namespace.

It is intended to serve as the basis for components in IEPD and EIEM schema documents, including subset, constraint, extension, and exchange schema documents.

See also reference schema document set.

reference schema document set A set of related reference schema documents, such as a NIEM release. See also reference schema document.

release A reference schema document set published by the NIEM Program Management Office (PMO) at http://release.niem.gov/ and assigned a unique version number. Each schema defines data components for use in NIEM information exchanges. Each release is independent of other releases, although a schema document may occur in multiple releases. A release is of high quality, and has been vetted by NIEM governance bodies. A numbered release may be a major, minor, or micro release.

schema document set coherence A schema document set is coherent when it has the following properties: (1) the set does not refer to a schema document outside the set (i.e., the set is closed), and (2) the set does not include two different versions of the same component in an incompatible way.

schema document subset An XML schema document set built from a reference schema document set by applying subset operations to that reference schema documents in that set. See also subset schema document.

subset schema document An XML schema document that meets all of the following criteria:

It is built from a referenc schema document set where one or more reference schema documents has been substituted by a its corresponding subset schema document.

It is built from a reference schema document by applying subset operations to the XML schema statements in a reference schema document.

It is explicitly designated as a subset schema document. This may be declared by an MPD catalog or by a tool-specific mechanism outside the subset schema document.

It has a target namespace previously defined by a reference schema document. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema document.

It does not alter the business semantics of components in its namespace. The reference schema document defines these business semantics.

It is intended to express the limited vocabulary necessary for an IEPD or EIEM and to support XML Schema validation for an IEPD.

See also schema document subset.

wantlist An XML document that represents a NIEM schema document subset. A NIEM wantlist identifies the data component requirements declared by the author of a subset; it does not identify the data component dependencies required to reconstitute the complete schema document subset. The complete schema document subset can be computed from the reference schema document or document set from which the subset was derived.

Appendix C. References

[FEA DRM]: *The Federal Enterprise Architecture Data Reference Model*, Version 1.0, September 2004. Available from `http://xml.gov/documents/completed/DRMv1.pdf`. A more recent DRM Version 2.0, 17 November 2005 is available from http://www.whitehouse.gov/omb/assets/egov_docs/DRM_2_0_Final.pdf

[GJXDM IEPD Guidelines]: *GJXDM Information Exchange Package Documentation Guidelines*, Version 1.1, Global XML Structure Task Force (GXSTF), 2 March 2005. Available from http://it.ojp.gov/documents/global_jxdm_IEPD_guidelines_v1_1.pdf

[ISO Schematron]: *Schema Definition Languages (DSDL)*, "Part 3: Rule-based validation : Schematron", ISO/IEC 19757-3:2006(E), First edition, 1 June 2006. Available from http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip.

[NIEM BIEC]: *Business Information Exchange Components (BIEC)*, Version 1.0, NIEM Technical Architecture Committee (NTAC), March 2011. Available from http://reference.niem.gov/niem/guidance/business-information-exchange-components/1.0/.

[NIEM Conformance]: *NIEM Conformance*, Version 1.0, NIEM Technical Architecture Committee (NTAC), 15 September 2008. Available from http://reference.niem.gov/niem/specification/conformance/1.0/.

[NIEM Concept of Operations]: *NIEM Concept of Operations*, Version 0.5, NIEM Program Management Office, 9 January 2007. Available from http://reference.niem.gov/niem/guidance/concept-of-operations/.

[NIEM Domain Update Specification]: *NIEM Domain Update Specification*, Version 1.0, NIEM Technical Architecture Committee (NTAC), 5 November 2010. Available from http://reference.niem.gov/niem/specification/domain-update/1.0/.

[NIEM High-Level Tool Architecture]: *NIEM High-Level Tool Architecture*, Version 1.1, NIEM Technical Architecture Committee, 1 December 2008. Available from http://reference.niem.gov/niem/specification/high-level-tool-architecture/1.1/.

[NIEM High-Level Version Architecture]: *NIEM High Level Version Architecture (HLVA)*, Version 1.0, NIEM Technical Architecture Committee, 2008. Available from http://reference.niem.gov/niem/specification/high-level-version-architecture/1.0/.

[NIEM IEPD Requirements]: *Requirements for a National Information Exchange Model (NIEM) Information Exchange Package Documentation (IEPD) Specification*, Version 2.1, June 2006. Available from http://reference.niem.gov/niem/guidance/iepd-requirements/2.1/.

[NIEM Implementation Guidance]: "NIEM Implementation Guide", NIEM Program Management Office. Available from https://www.niem.gov/program-managers/Pages/implementation-guide.aspx.

[NIEM Introduction]: *Introduction to the National Information Exchange Model (NIEM)*, Version 0.3, NIEM Program Management Office, 12 February 2007. Available from http://reference.niem.gov/niem/guidance/introduction/.

[NIEM NDR]: *NIEM Naming and Design Rules (NDR)*, Version 3.0, NIEM Technical Architecture Committee (NTAC), 31 October 2008. Available from `http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/`.

[NIEM SSGT]: NIEM Schema Subset Generation Tool (SSGT). Available from `http://tools.niem.gov/niemtools/ssgt/index.iepd`.

[NIEM User Guide]: *NIEM User Guide*, Volume 1, U.S. Department of Justice, Office of Justice Programs, (date unknown). Available from `http://reference.niem.gov/niem/guidance/user-guide/vol1/`.

[RFC2119 Key Words]: Bradner, S., *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997. Available from `http://www.ietf.org/rfc/rfc2119.txt`.

[RFC3986 URI]: Berners-Lee, T., et al., *Uniform Resource Identifier (URI): Generic Syntax*, Request for Comments 3986, Network Working Group, January 2005. Available from `http://tools.ietf.org/html/rfc3986`.

[W3-EXI]: *Efficient XML Interchange (EXI) Format*, Version 1.0, W3C Recommendation, 10 March 2011. Available from `http://www.w3.org/TR/2011/REC-exi-20110310/`.

[W3-OWL]: *OWL Web Ontology Language Reference*, W3C Recommendation 10 February 2004. Available from `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

[W3-RDF]: *Resource Description Framework (RDF): Concepts and Abstract Syntax*, W3C Recommendation 10 February 2004. Available from `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[W3-XML]: *Extensible Markup Language (XML)*, Version 1.0, Fifth Edition, W3C Recommendation 26 November 2008. Available from `http://www.w3.org/TR/2008/REC-xml-20081126/`.

[W3-XML-InfoSet]: *XML Information Set*, Second Edition, W3C Recommendation 4 February 2004. Available from `http://www.w3.org/TR/2004/REC-xml-infoset-20040204/`.

[W3-XML-Namespaces]: *Namespaces in XML*, Second Edition, World Wide Web Consortium 16 August 2006. Available from `http://www.w3.org/TR/2006/REC-xml-names-20060816/`.

[W3 XML Schema Datatypes]: *XML Schema Part 2: Datatypes*, Second Edition, W3C Recommendation 28 October 2004. Available from `http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/`.

[W3 XML Schema Structures]: *XML Schema Part 1: Structures*, Second Edition, W3C Recommendation 28 October 2004. Available from `http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`.

[W3-XSLT]: *XSL Transformations (XSLT)*, Version 1.0, W3C Recommendation 16 November 1999. Available from `http://www.w3.org/TR/1999/REC-xslt-19991116`.

[W3-XSLT2]: *XSL Transformations (XSLT)*, Version 2.0, W3C Recommendation 23 January

2007. Available from http://www.w3.org/TR/2007/REC-xslt20-20070123/.

[PKZIP]: *APPNOTE.TXT - .ZIP File Format Specification*, Version: 6.3.2, Revised: 28 September 2007, Copyright (c) 1989 - 2007 PKWare Inc. Available from http://www.pkware.com/documents/casestudies/APPNOTE.TXT.

Appendix D. Footnotes

A NIEM-conformant IEPD or EIEM is required to have at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

An IEPD or EIEM change log may be informal in nature and is not required to conform to the strict XML schema definition that is mandatory for releases, core updates, and domain updates. Therefore, csv or TEXT formats are also authorized.

Each NIEM release, core update, and domain update is required to conform to the change log XML schema in http://reference.niem.gov/niem/resource/mpd/changelog/.